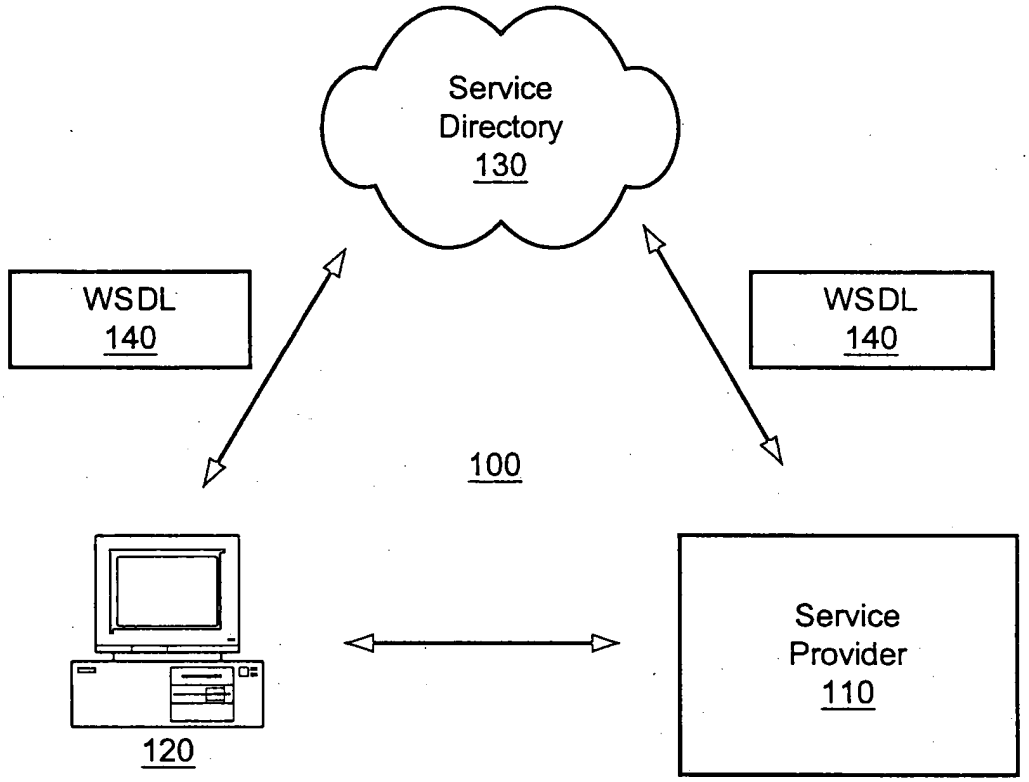


FIG. 1
(Prior Art)

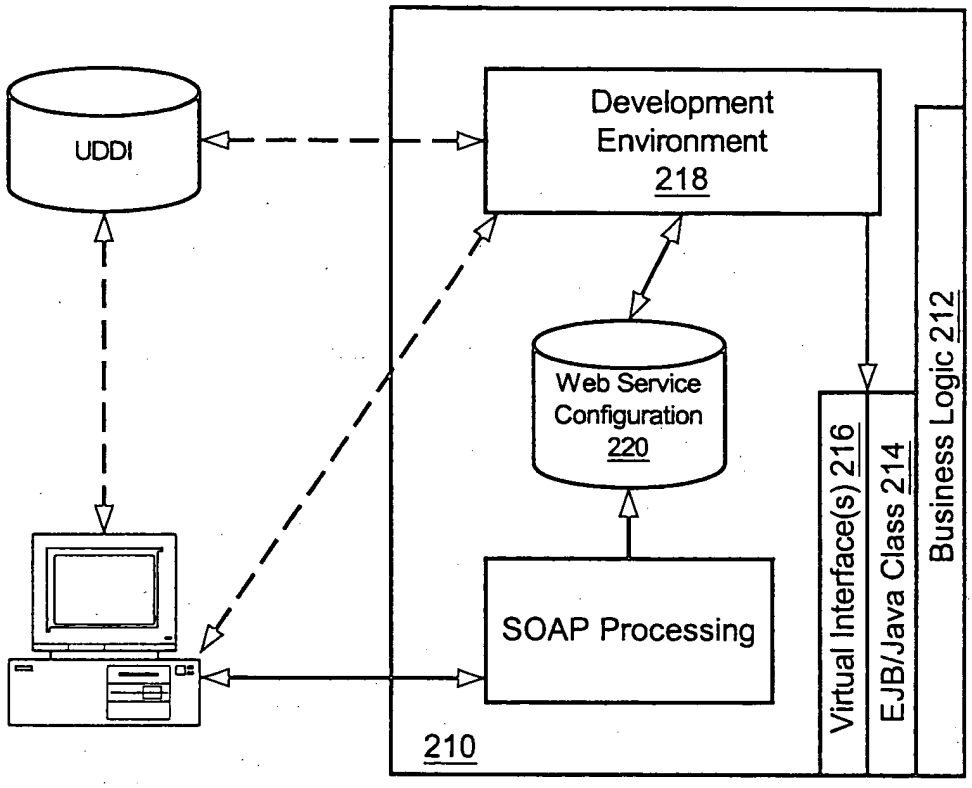

UDDI
Development Environment 218
Web Service Configuration 220
SOAP Processing
Virtual Interface(s) 216
EJB/Java Class 214
Business Logic 212
210

FIG. 2

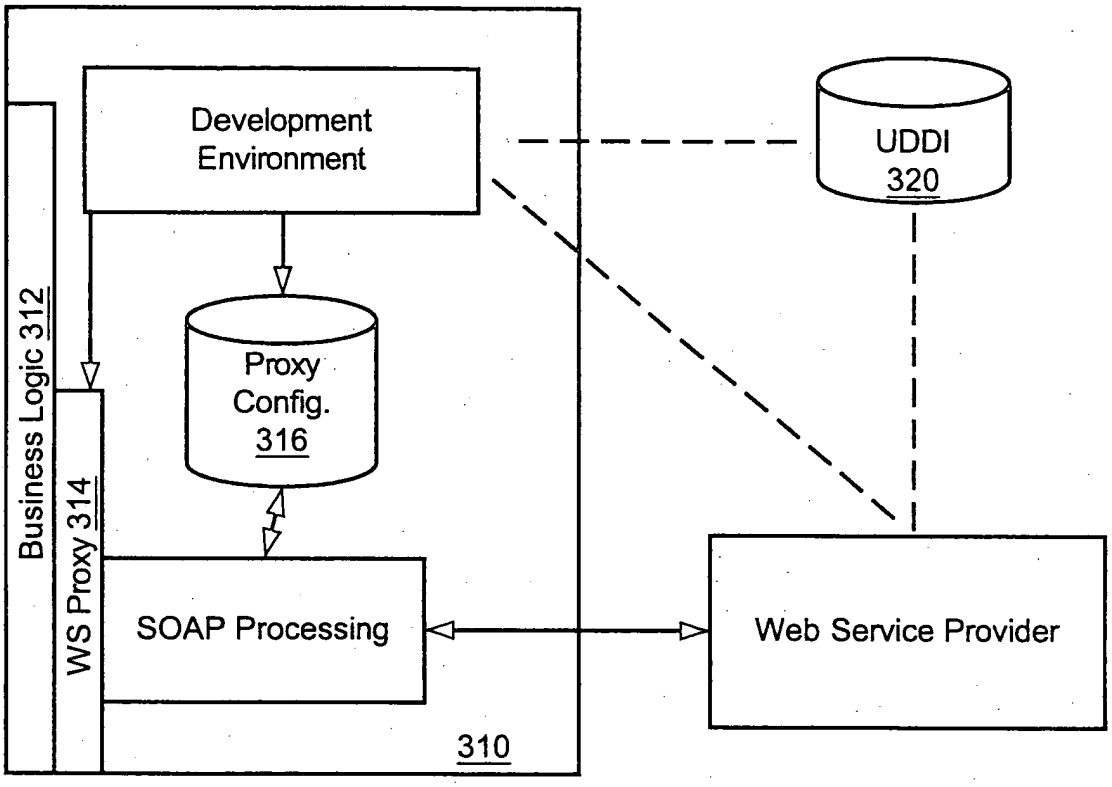

Development Environment
UDDI 320
Business Logic 312
Proxy Config. 316
WS Proxy 314
SOAP Processing
Web Service Provider
310

FIG. 3

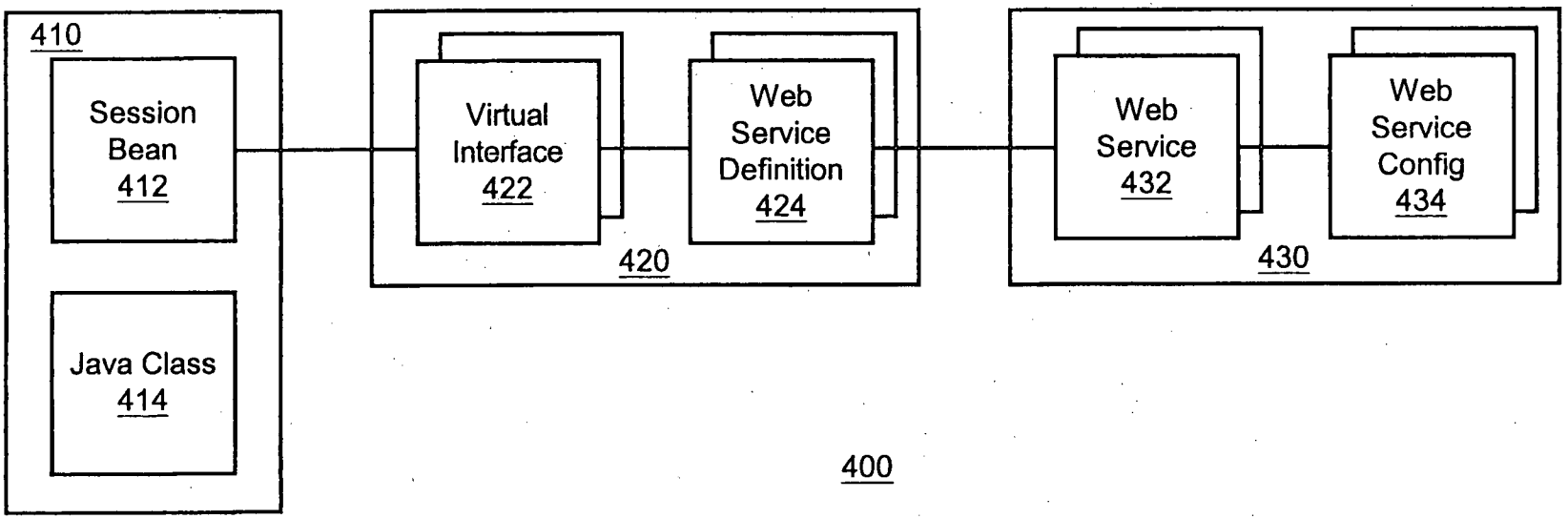

410
Session Bean 412
Java Class 414
Virtual Interface 422
Web Service Definition 424
420
Web Service 432
Web Service Config 434
430
400

FIG. 4

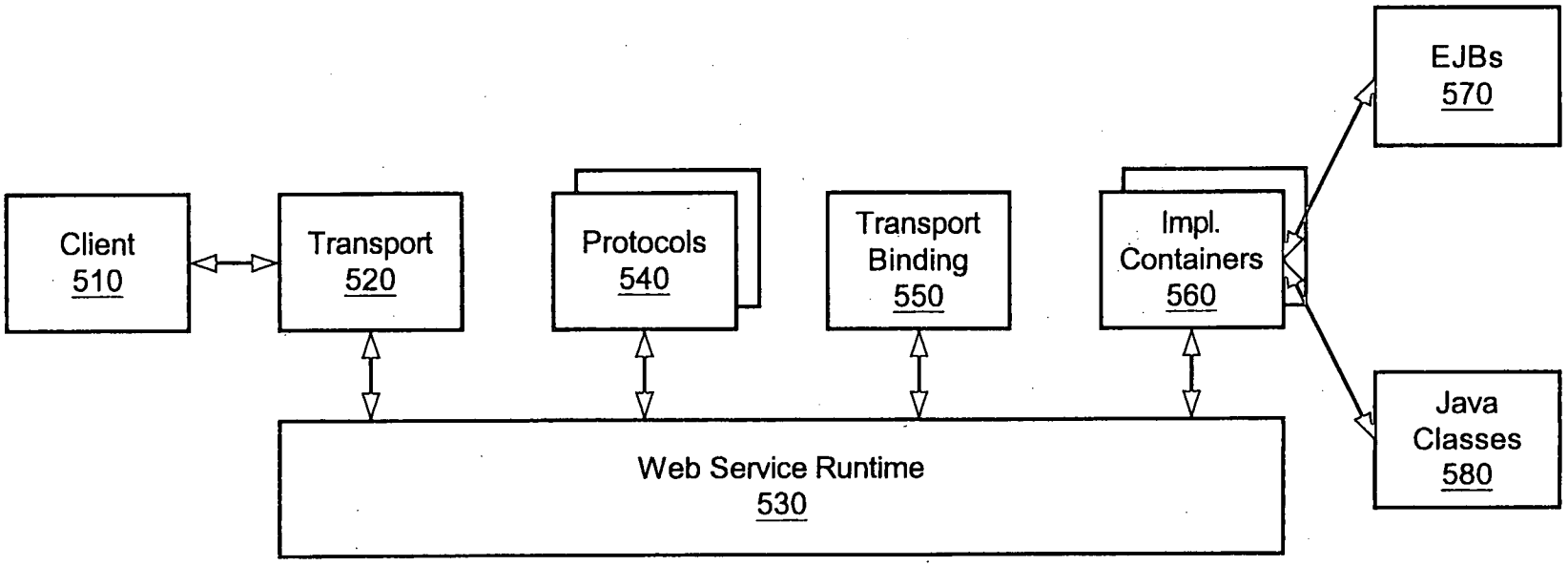

Client
510
Transport
520
Protocols
540
Transport
Binding
550
Impl.
Containers
560
EJBs
570
Java
Classes
580
Web Service Runtime
530
500

FIG. 5

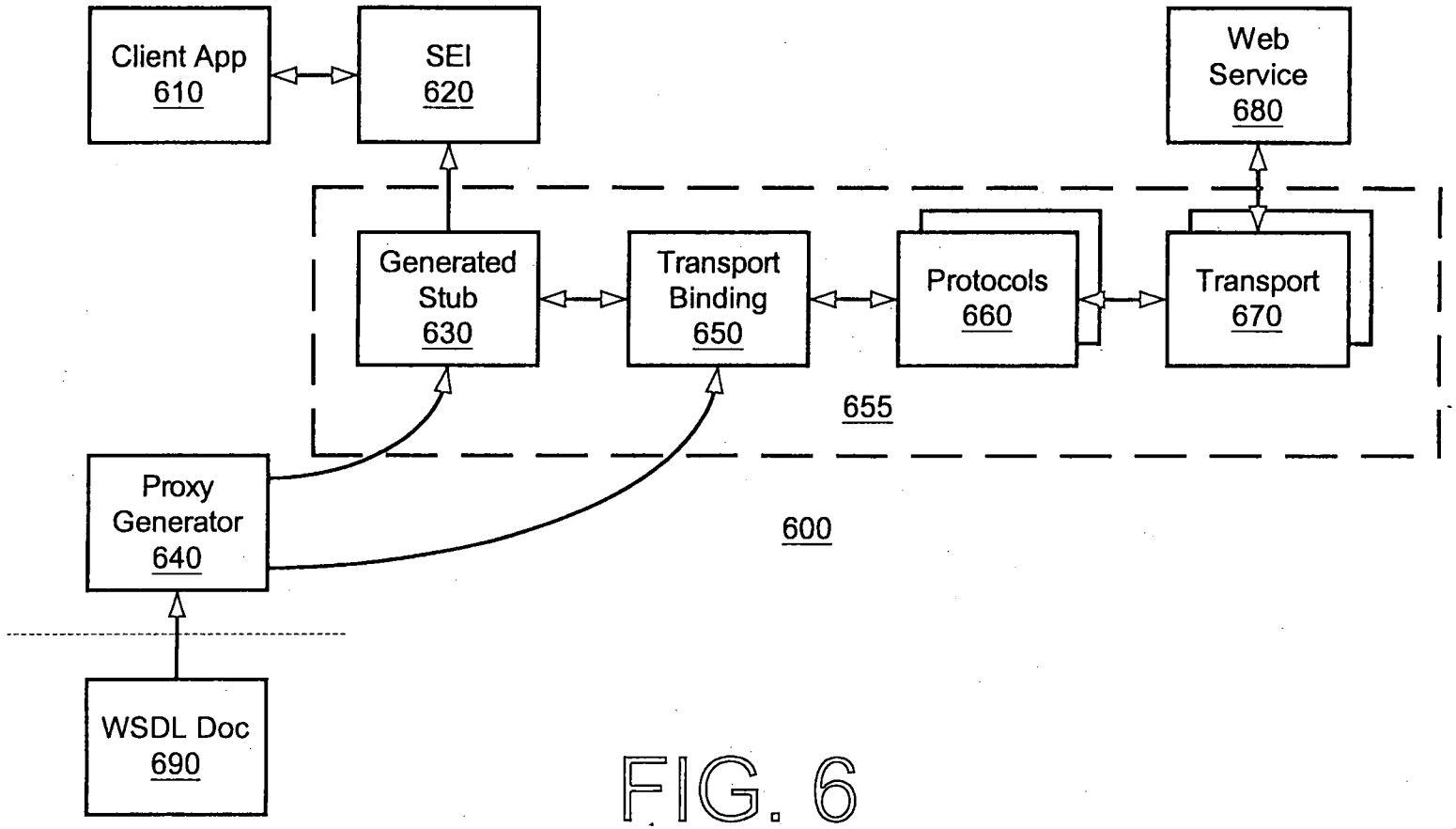

Client App
610
SEI
620
Web
Service
680
Generated
Stub
630
Transport
Binding
650
Protocols
660
Transport
670
655
Proxy
Generator
640
600
WSDL Doc
690

FIG. 6

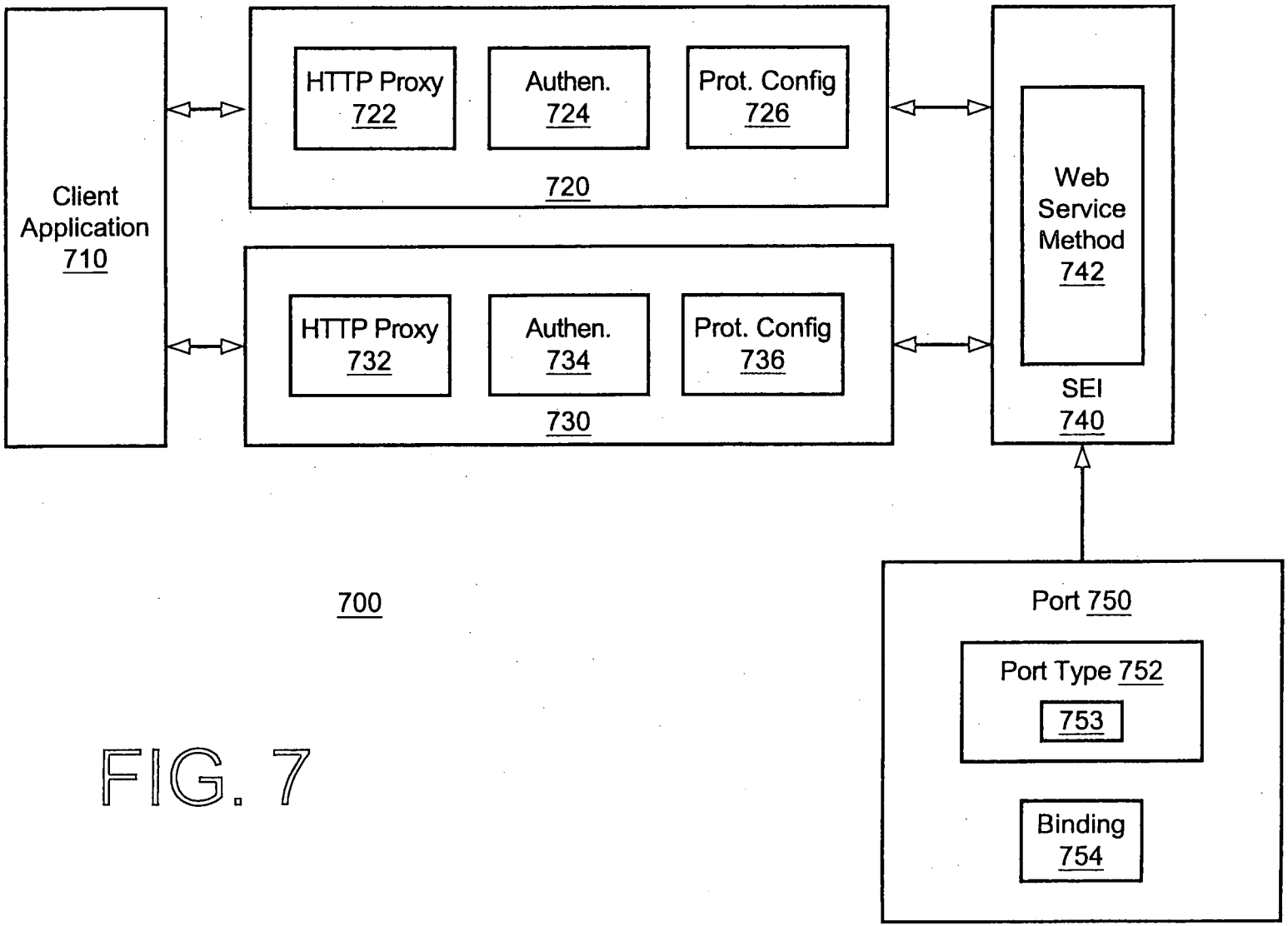

Client Application 710
HTTP Proxy 722
Authen. 724
Prot. Config 726
720
HTTP Proxy 732
Authen. 734
Prot. Config 736
730
Web Service Method 742
SEI 740
700
Port 750
Port Type 752
753
Binding 754

FIG. 7

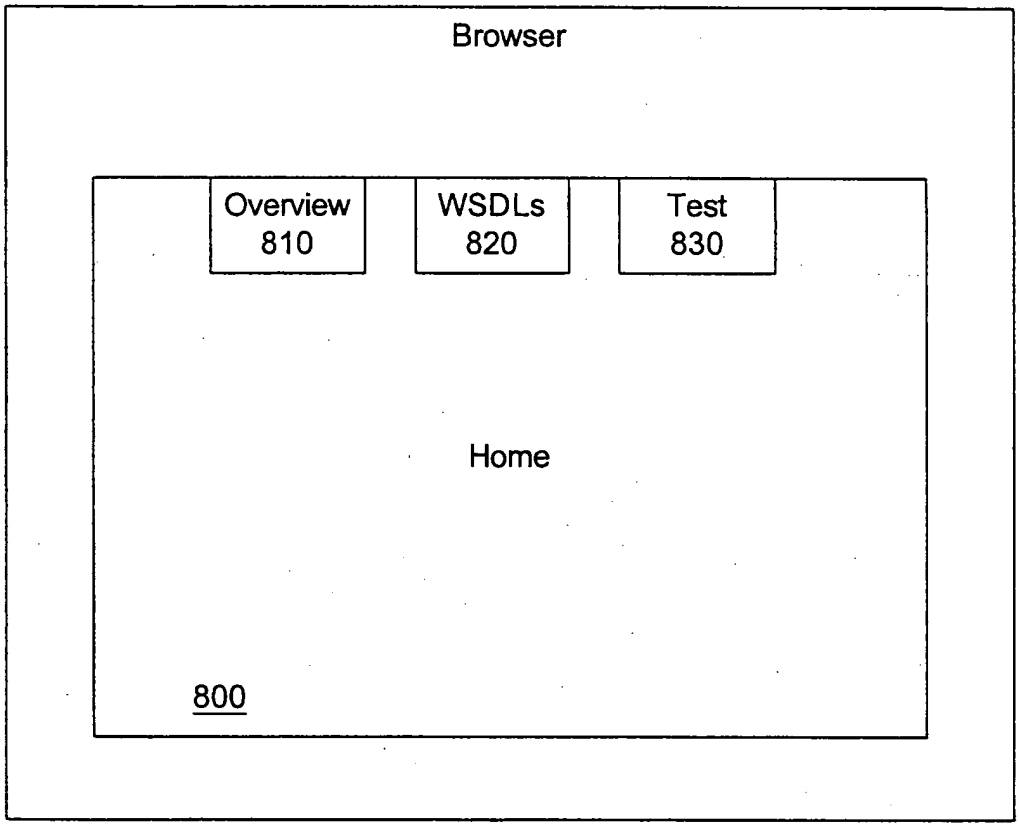


FIG. 8

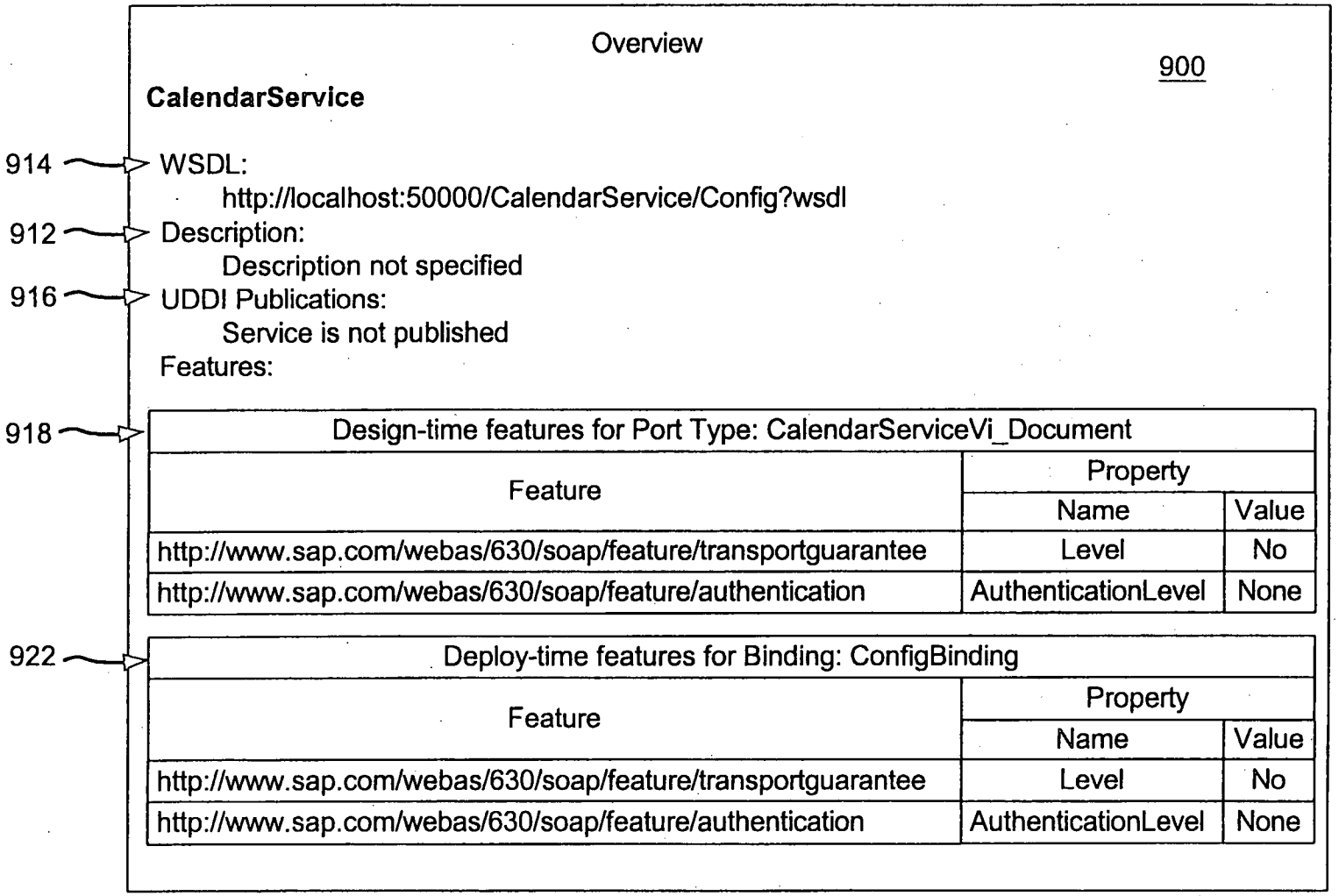

Overview
900
CalendarService
914
WSDL:
http://localhost:50000/CalendarService/Config?wsdl
912
Description:
Description not specified
916
UDDI Publications:
Service is not published
Features:
918
Design-time features for Port Type: CalendarServiceVi_Document
Feature
Property
Name
Value
http://www.sap.com/webas/630/soap/feature/transportguarantee
Level
No
http://www.sap.com/webas/630/soap/feature/authentication
AuthenticationLevel
None
922
Deploy-time features for Binding: ConfigBinding
Feature
Property
Name
Value
http://www.sap.com/webas/630/soap/feature/transportguarantee
Level
No
http://www.sap.com/webas/630/soap/feature/authentication
AuthenticationLevel
None

FIG. 9

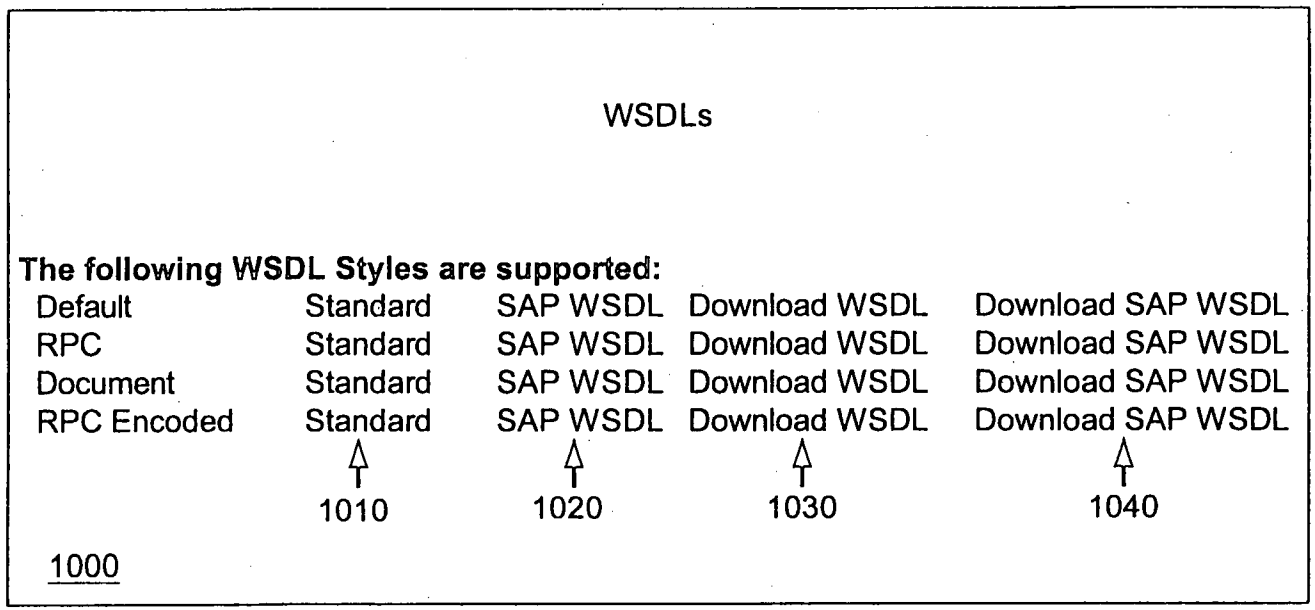

WSDLs
The following WSDL Styles are supported:
Default Standard SAP WSDL Download WSDL Download SAP WSDL
RPC Standard SAP WSDL Download WSDL Download SAP WSDL
Document Standard SAP WSDL Download WSDL Download SAP WSDL
RPC Encoded Standard SAP WSDL Download WSDL Download SAP WSDL
1010
1020
1030
1040
1000

FIG. 10

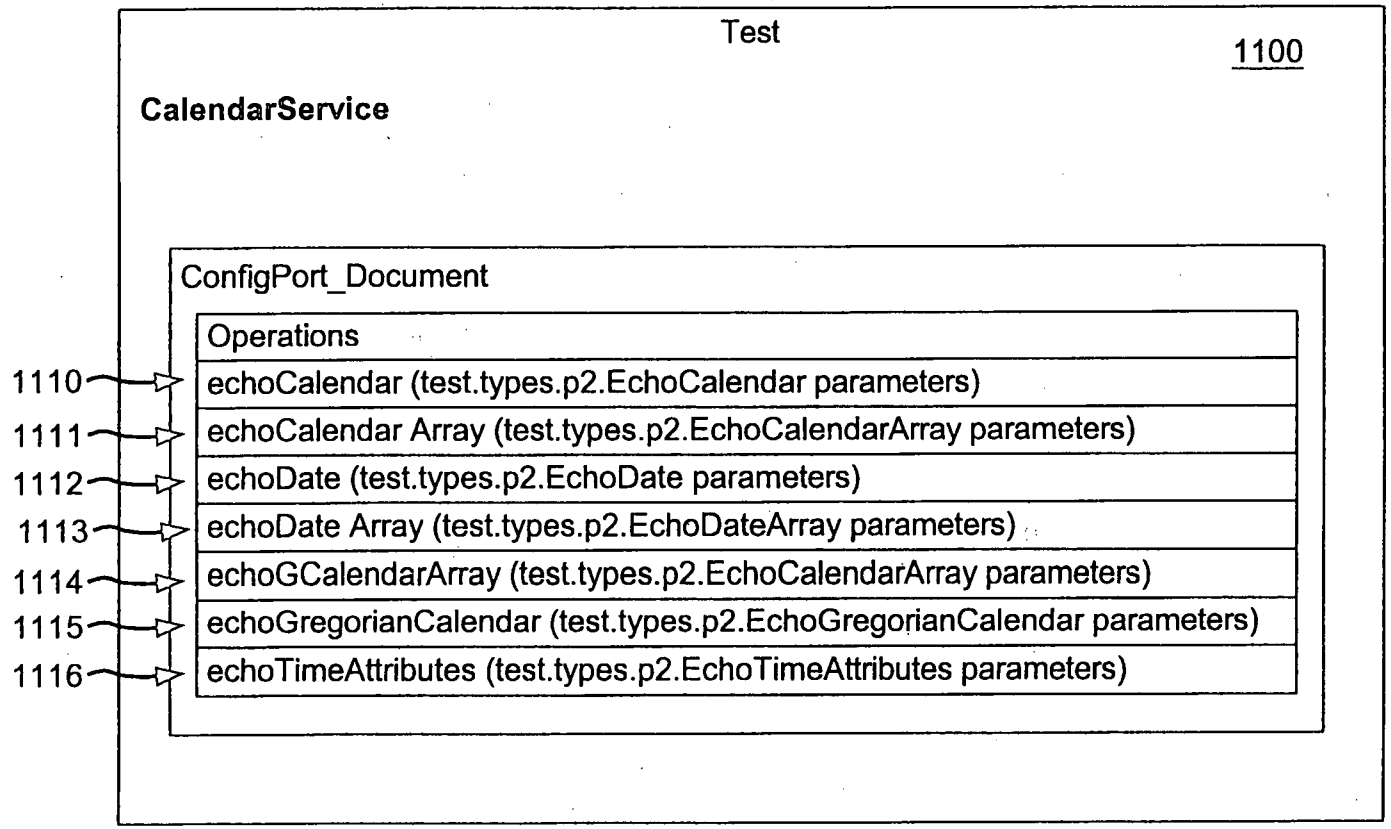

Test
1100
CalendarService
ConfigPort_Document
Operations
1110
echoCalendar (test.types.p2.EchoCalendar parameters)
1111
echoCalendar Array (test.types.p2.EchoCalendarArray parameters)
1112
echoDate (test.types.p2.EchoDate parameters)
1113
echoDate Array (test.types.p2.EchoDateArray parameters)
1114
echoGCalendarArray (test.types.p2.EchoCalendarArray parameters)
1115
echoGregorianCalendar (test.types.p2.EchoGregorianCalendar parameters)
1116
echoTimeAttributes (test.types.p2.EchoTimeAttributes parameters)

FIG. 11

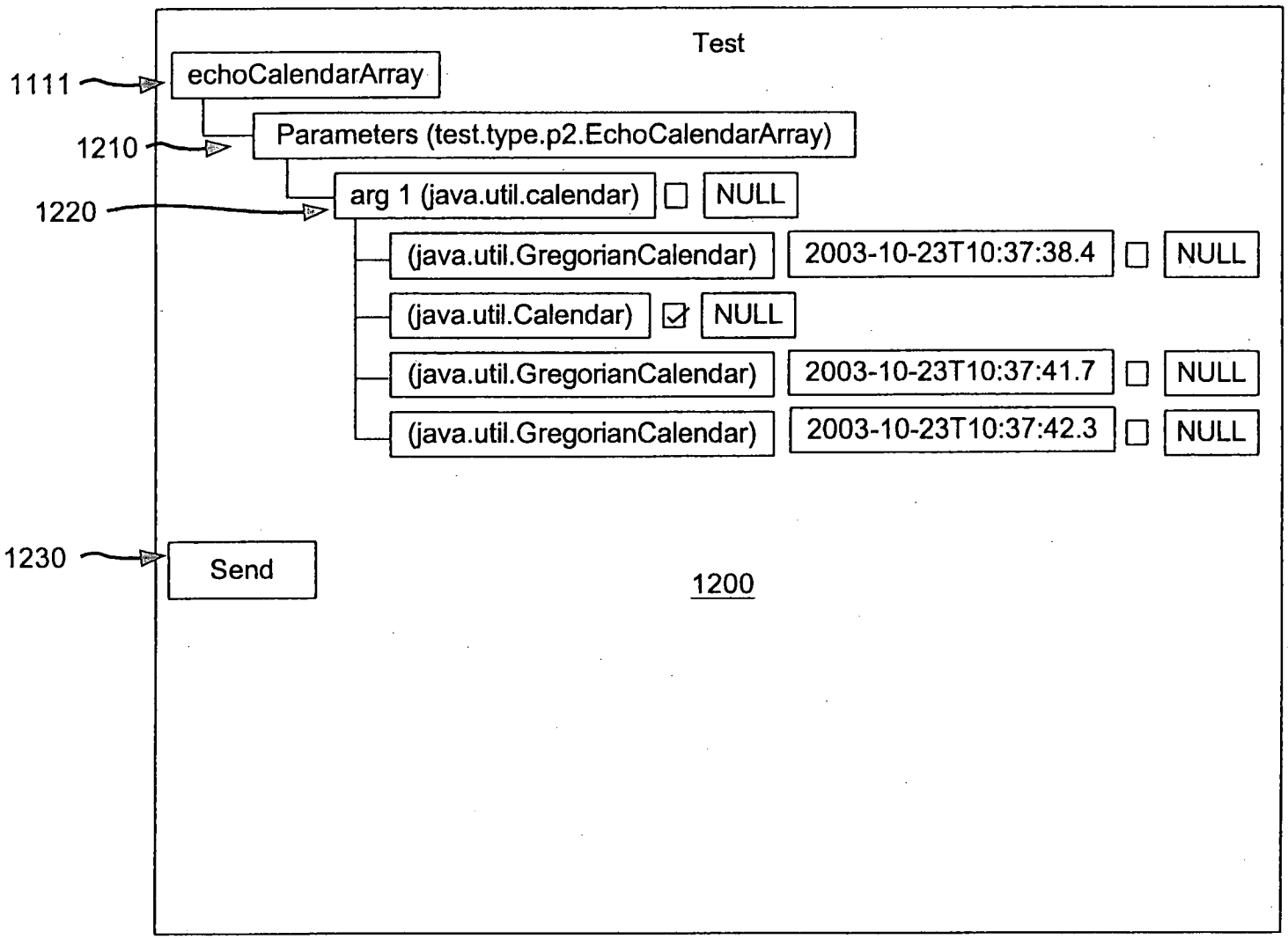

Test
1111
echoCalendarArray
1210
Parameters (test.type.p2.EchoCalendarArray)
1220
arg 1 (java.util.calendar)
NULL
(java.util.GregorianCalendar)
2003-10-23T10:37:38.4
NULL
(java.util.Calendar)
NULL
(java.util.GregorianCalendar)
2003-10-23T10:37:41.7
NULL
(java.util.GregorianCalendar)
2003-10-23T10:37:42.3
NULL
1230
Send
1200

FIG. 12

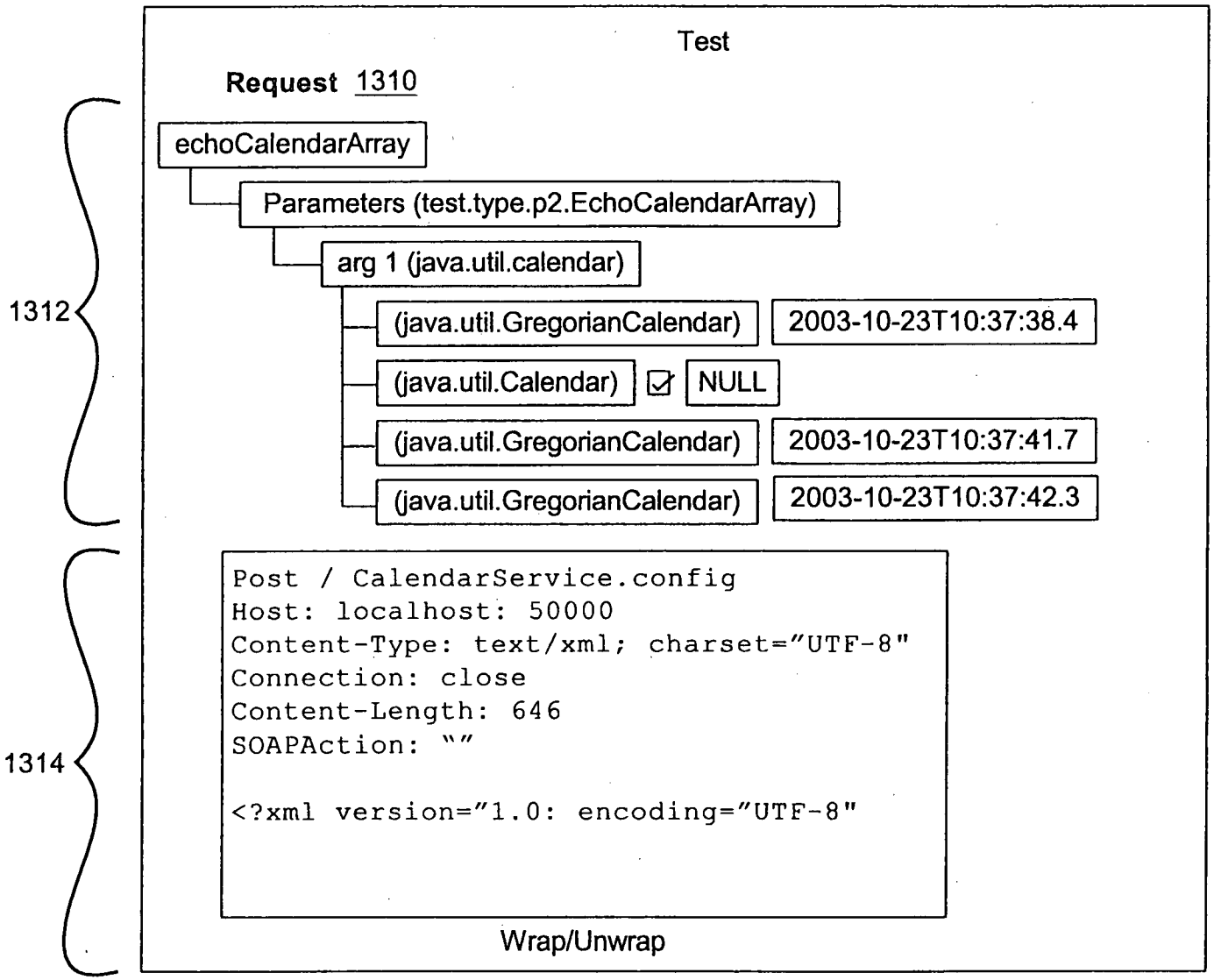

Test
Request 1310
echoCalendarArray
Parameters (test.type.p2.EchoCalendarArray)
arg 1 (java.util.calendar)
(java.util.GregorianCalendar)
2003-10-23T10:37:38.4
(java.util.Calendar)
NULL
(java.util.GregorianCalendar)
2003-10-23T10:37:41.7
(java.util.GregorianCalendar)
2003-10-23T10:37:42.3
1312
Post / CalendarService.config
Host: localhost: 50000
Content-Type: text/xml; charset="UTF-8"
Connection: close
Content-Length: 646
SOAPAction: ""
<?xml version="1.0: encoding="UTF-8"
1314
Wrap/Unwrap

FIG. 13A

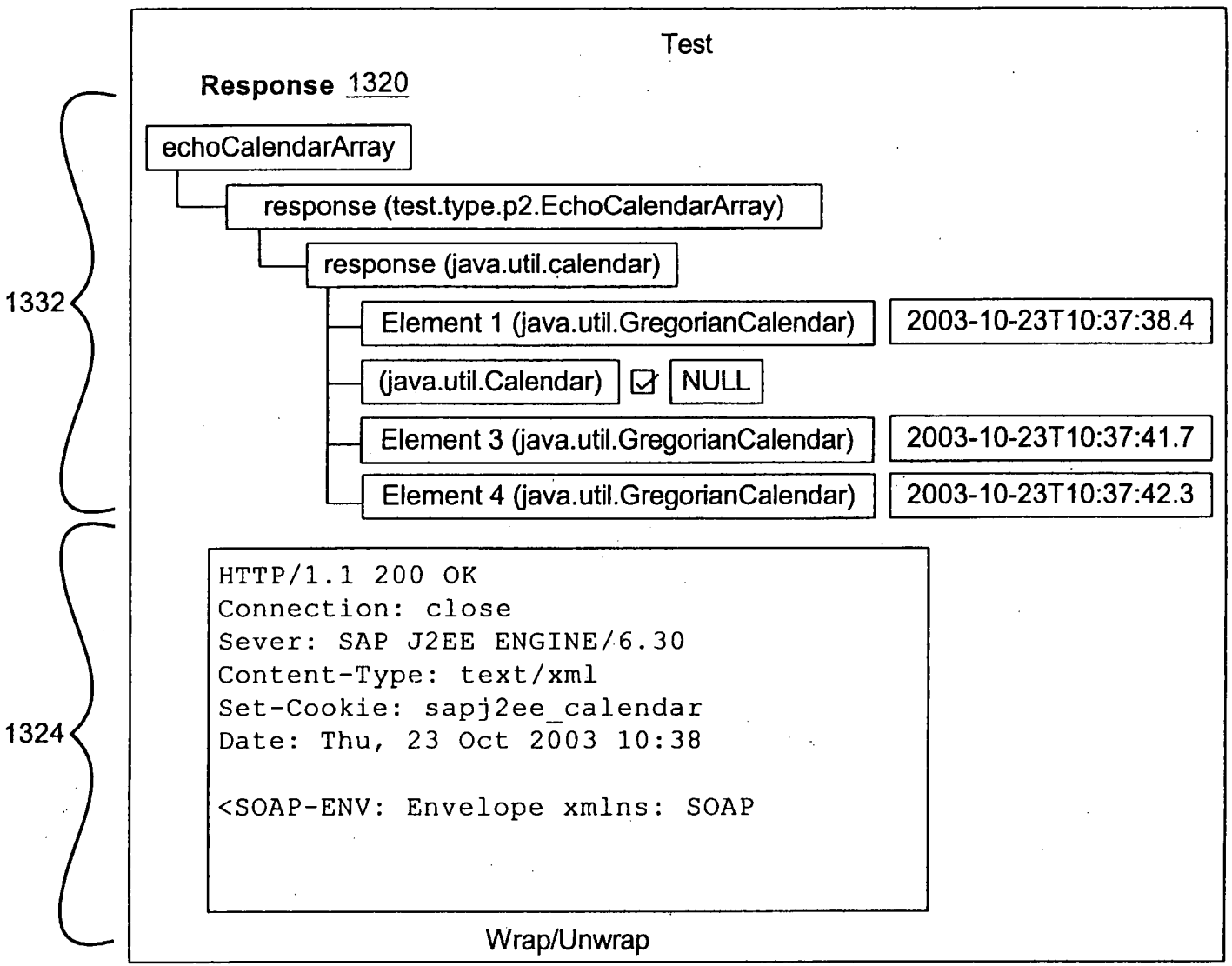

Test
Response 1320
echoCalendarArray
response (test.type.p2.EchoCalendarArray)
response (java.util.calendar)
Element 1 (java.util.GregorianCalendar)
2003-10-23T10:37:38.4
(java.util.Calendar)
NULL
Element 3 (java.util.GregorianCalendar)
2003-10-23T10:37:41.7
Element 4 (java.util.GregorianCalendar)
2003-10-23T10:37:42.3
1332
HTTP/1.1 200 OK
Connection: close
Sever: SAP J2EE ENGINE/6.30
Content-Type: text/xml
Set-Cookie: sapj2ee_calendar
Date: Thu, 23 Oct 2003 10:38
<SOAP-ENV: Envelope xmlns: SOAP
1324
Wrap/Unwrap

FIG. 13B

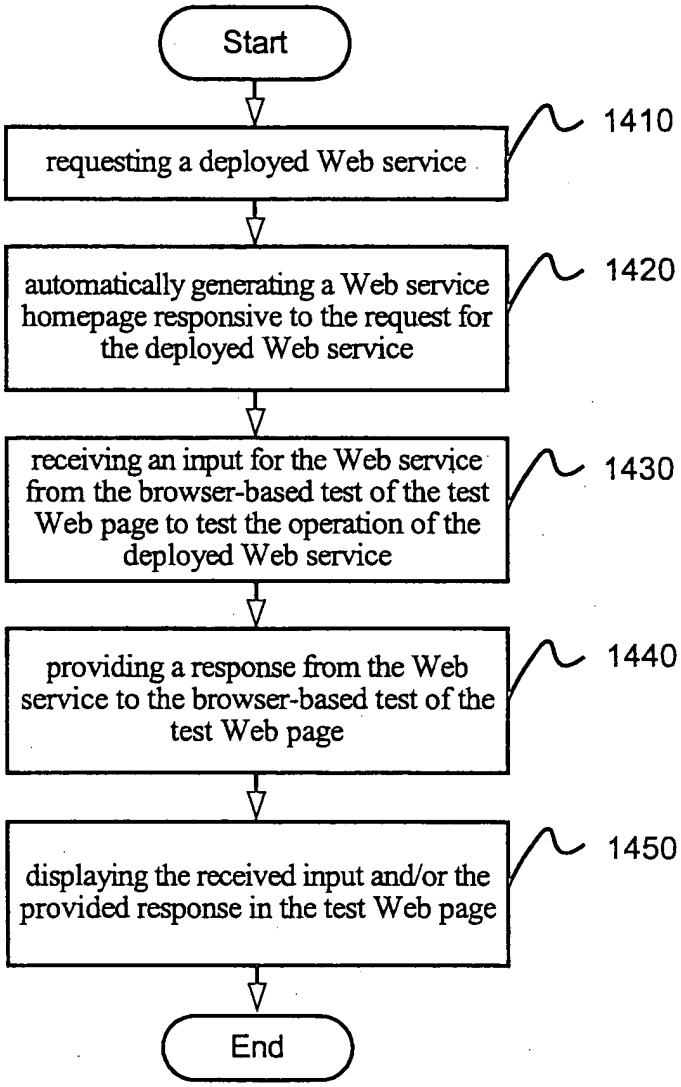

Start
requesting a deployed Web service
1410
automatically generating a Web service homepage responsive to the request for the deployed Web service
1420
receiving an input for the Web service from the browser-based test of the test Web page to test the operation of the deployed Web service
1430
providing a response from the Web service to the browser-based test of the test Web page
1440
displaying the received input and/or the provided response in the test Web page
1450
End

FIG. 14

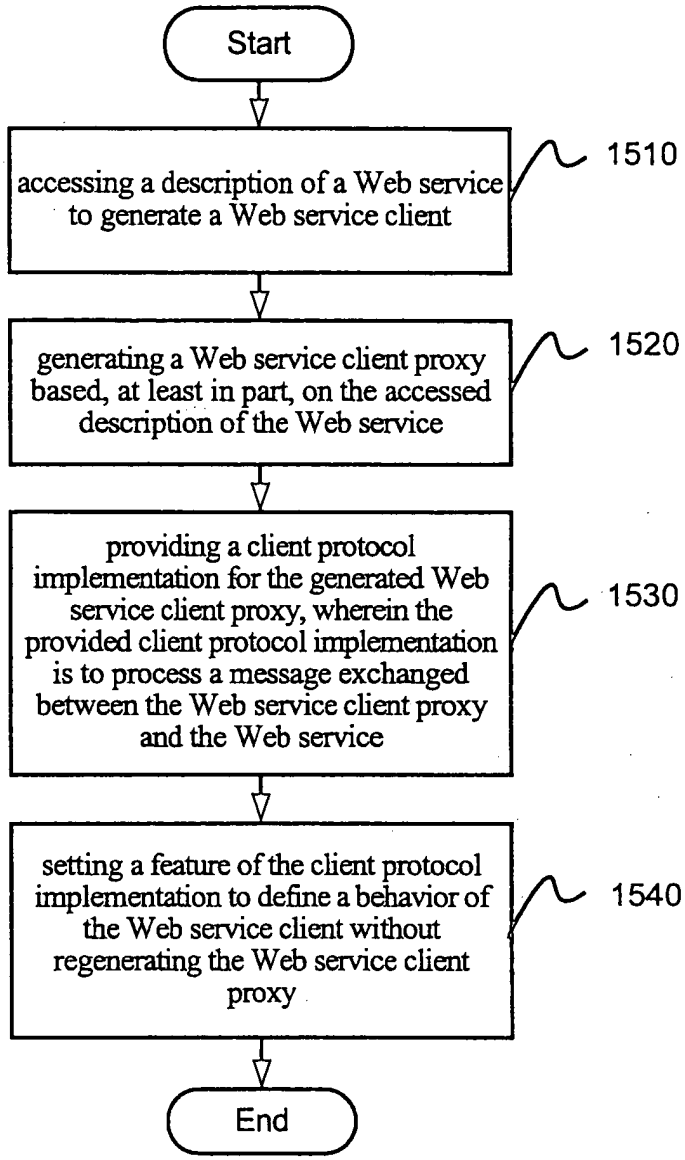

Start
accessing a description of a Web service to generate a Web service client
1510
generating a Web service client proxy based, at least in part, on the accessed description of the Web service
1520
providing a client protocol implementation for the generated Web service client proxy, wherein the provided client protocol implementation is to process a message exchanged between the Web service client proxy and the Web service
1530
setting a feature of the client protocol implementation to define a behavior of the Web service client without regenerating the Web service client proxy
1540
End

FIG. 15

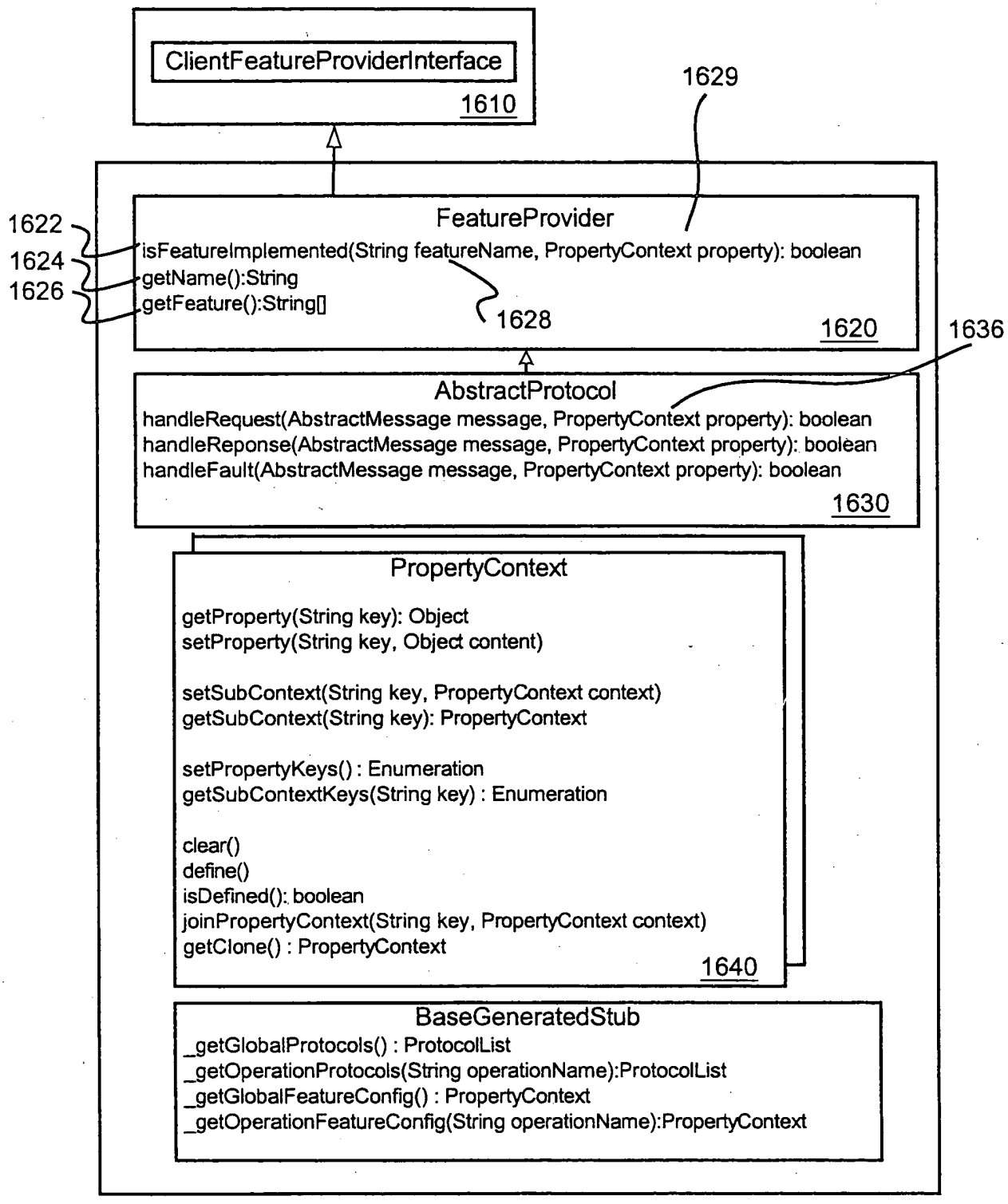

ClientFeatureProviderInterface
1610
1629
FeatureProvider
1622
1624
1626
isFeatureImplemented(String featureName, PropertyContext property): boolean
getName():String
getFeature():String[]
1628
1620
1636
AbstractProtocol
handleRequest(AbstractMessage message, PropertyContext property): boolean
handleReponse(AbstractMessage message, PropertyContext property): boolean
handleFault(AbstractMessage message, PropertyContext property): boolean
1630
PropertyContext
getProperty(String key): Object
setProperty(String key, Object content)
setSubContext(String key, PropertyContext context)
getSubContext(String key): PropertyContext
setPropertyKeys() : Enumeration
getSubContextKeys(String key) : Enumeration
clear()
define()
isDefined(): boolean
joinPropertyContext(String key, PropertyContext context)
getClone() : PropertyContext
1640
BaseGeneratedStub
_getGlobalProtocols() : ProtocolList
_getOperationProtocols(String operationName):ProtocolList
_getGlobalFeatureConfig() : PropertyContext
_getOperationFeatureConfig(String operationName):PropertyContext

FIG. 16

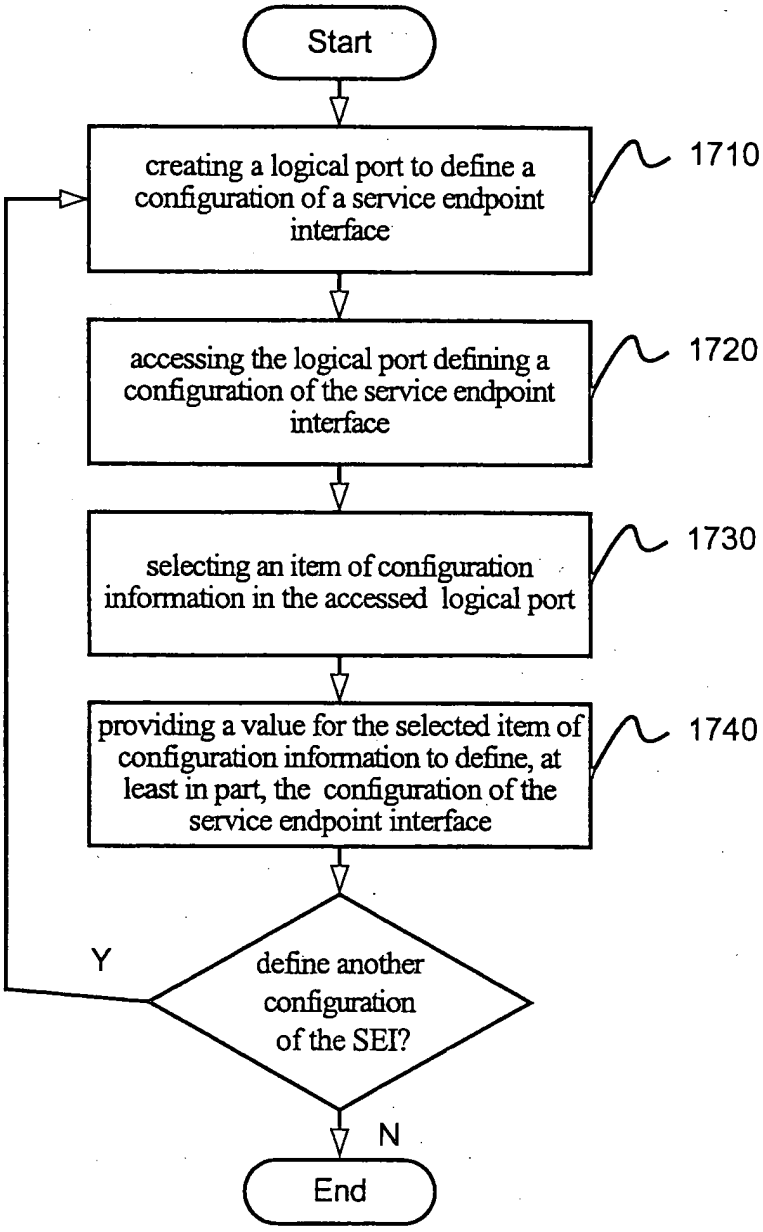

Start
creating a logical port to define a configuration of a service endpoint interface
1710
accessing the logical port defining a configuration of the service endpoint interface
1720
selecting an item of configuration information in the accessed logical port
1730
providing a value for the selected item of configuration information to define, at least in part, the configuration of the service endpoint interface
1740
define another configuration of the SEI?
Y
N
End

FIG. 17

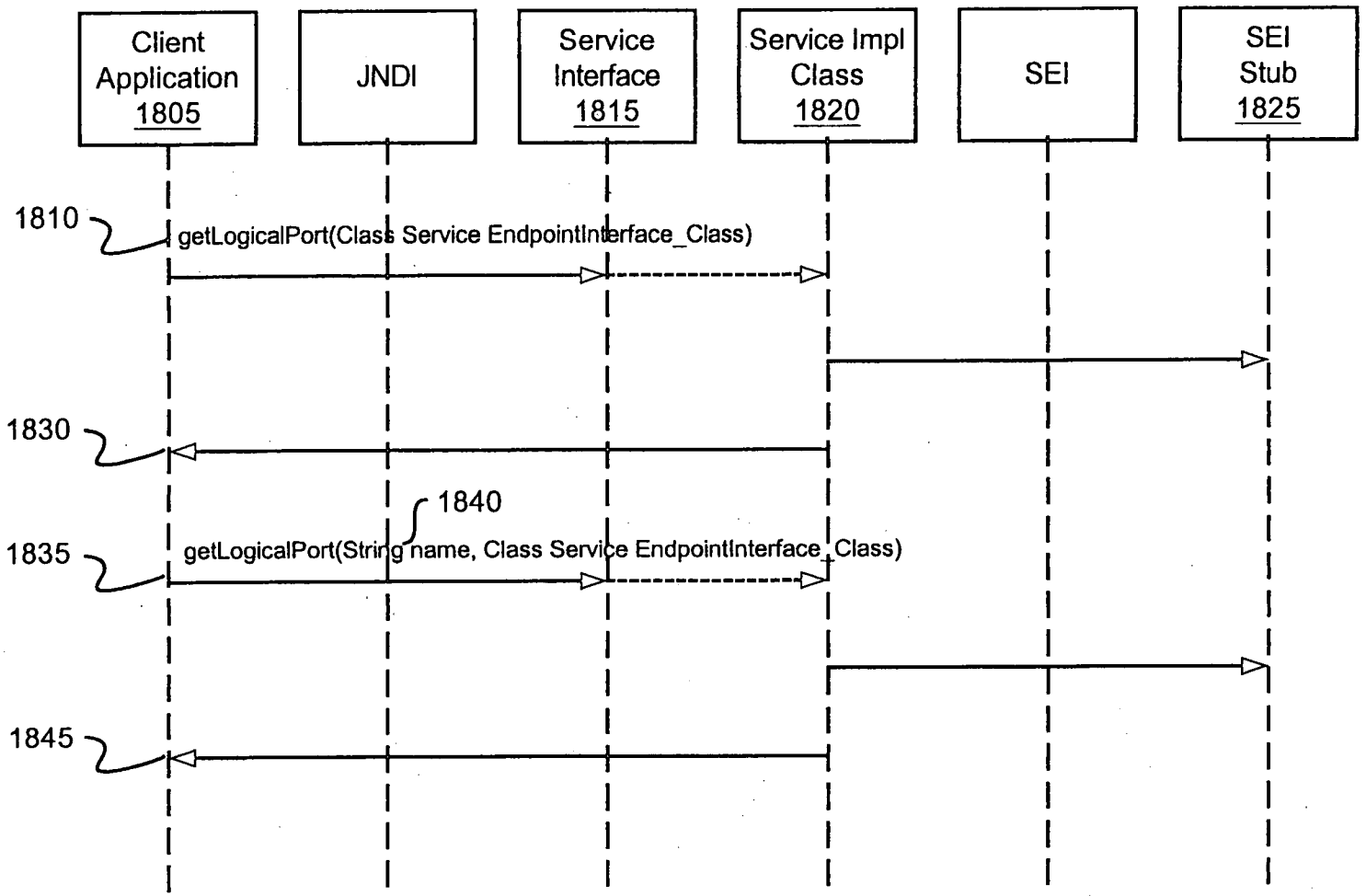

Client Application 1805
JNDI
Service Interface 1815
Service Impl Class 1820
SEI
SEI Stub 1825
1810
getLogicalPort(Class Service EndpointInterface_Class)
1830
1840
1835
getLogicalPort(String name, Class Service EndpointInterface_Class)
1845

FIG. 18

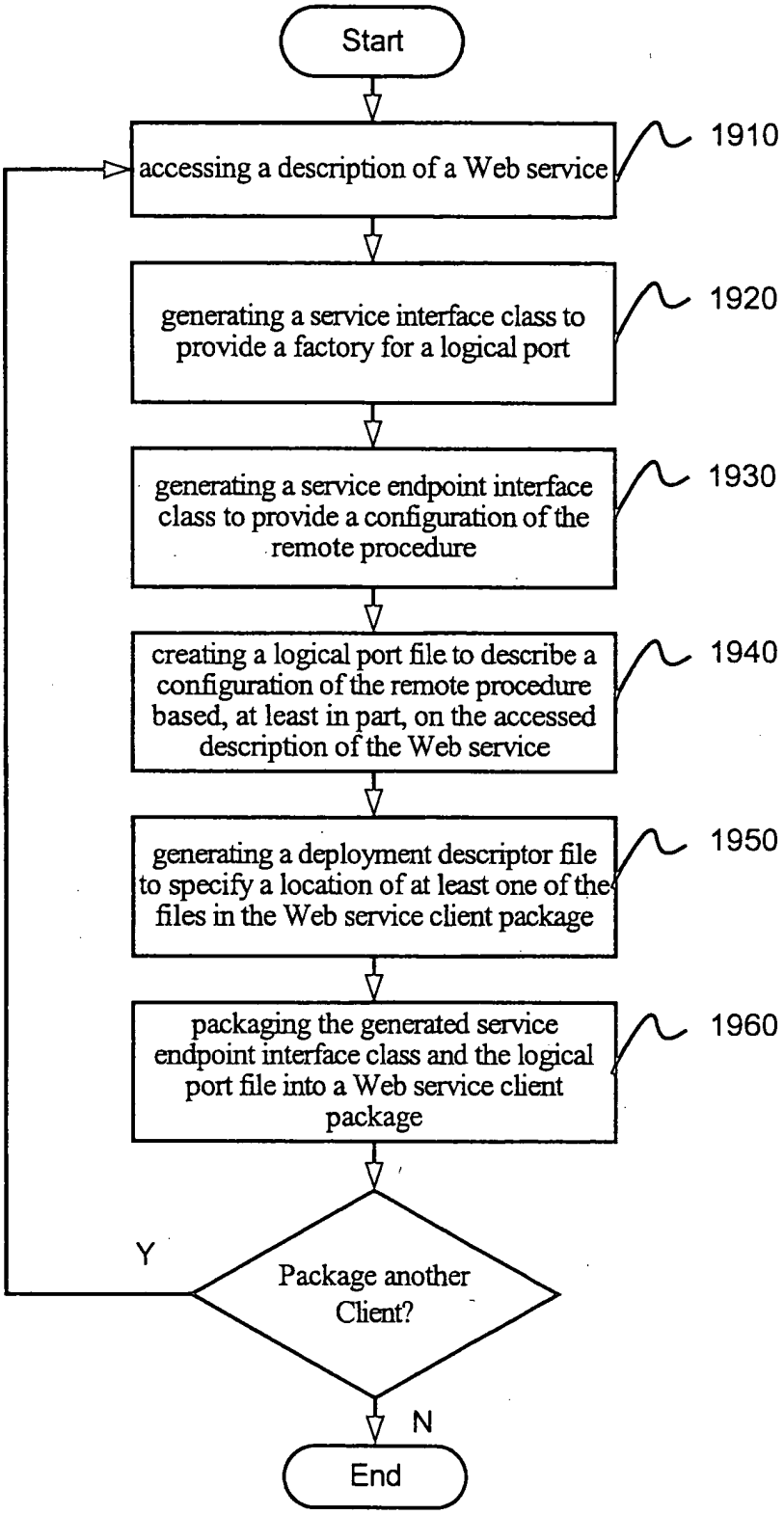

Start
accessing a description of a Web service
1910
generating a service interface class to provide a factory for a logical port
1920
generating a service endpoint interface class to provide a configuration of the remote procedure
1930
creating a logical port file to describe a configuration of the remote procedure based, at least in part, on the accessed description of the Web service
1940
generating a deployment descriptor file to specify a location of at least one of the files in the Web service client package
1950
packaging the generated service endpoint interface class and the logical port file into a Web service client package
1960
Package another Client?
Y
N
End

FIG. 19

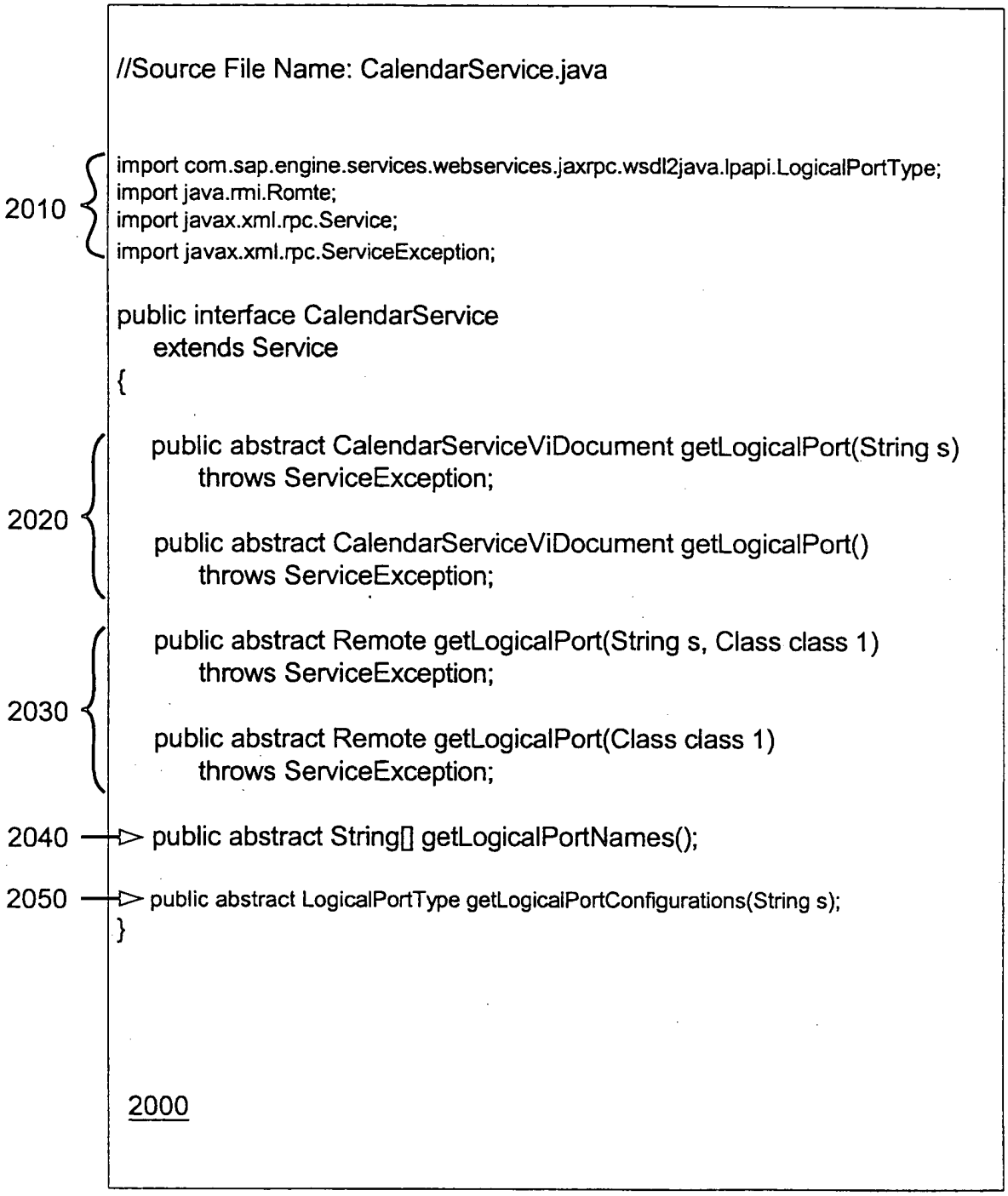

//Source File Name: CalendarService.java
2010
import com.sap.engine.services.webservices.jaxrpc.wsdl2java.lpapi.LogicalPortType;
import java.rmi.Romte;
import javax.xml.rpc.Service;
import javax.xml.rpc.ServiceException;
public interface CalendarService
extends Service
{
2020
public abstract CalendarServiceViDocument getLogicalPort(String s)
throws ServiceException;
public abstract CalendarServiceViDocument getLogicalPort()
throws ServiceException;
2030
public abstract Remote getLogicalPort(String s, Class class 1)
throws ServiceException;
public abstract Remote getLogicalPort(Class class 1)
throws ServiceException;
2040
public abstract String[] getLogicalPortNames();
2050
public abstract LogicalPortType getLogicalPortConfigurations(String s);
}
2000

FIG. 20

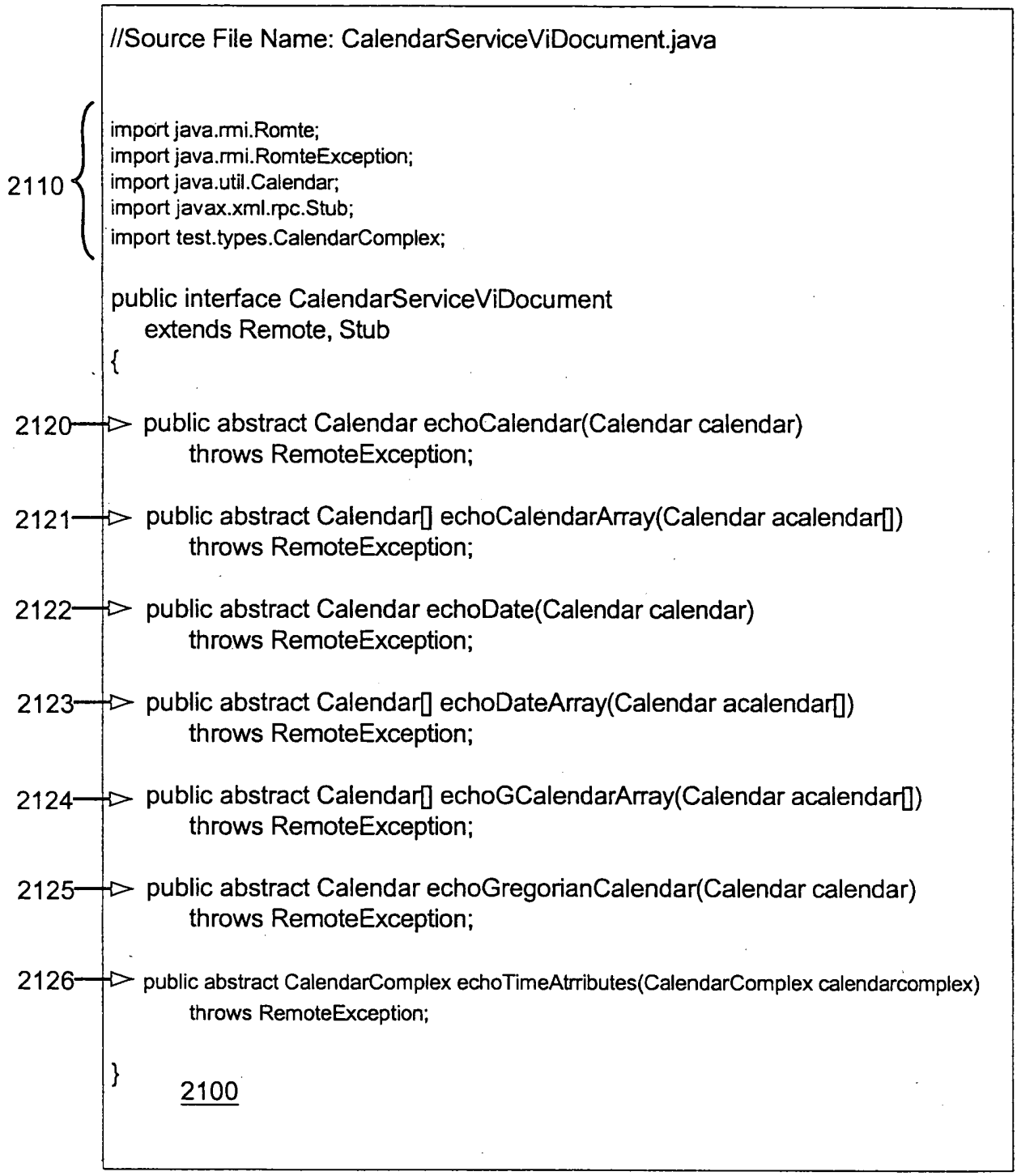

//Source File Name: CalendarServiceViDocument.java
2110
import java.rmi.Romte;
import java.rmi.RomteException;
import java.util.Calendar;
import javax.xml.rpc.Stub;
import test.types.CalendarComplex;
public interface CalendarServiceViDocument
extends Remote, Stub
{
2120
public abstract Calendar echoCalendar(Calendar calendar)
throws RemoteException;
2121
public abstract Calendar[] echoCalendarArray(Calendar acalendar[])
throws RemoteException;
2122
public abstract Calendar echoDate(Calendar calendar)
throws RemoteException;
2123
public abstract Calendar[] echoDateArray(Calendar acalendar[])
throws RemoteException;
2124
public abstract Calendar[] echoGCalendarArray(Calendar acalendar[])
throws RemoteException;
2125
public abstract Calendar echoGregorianCalendar(Calendar calendar)
throws RemoteException;
2126
public abstract CalendarComplex echoTimeAtrributes(CalendarComplex calendarcomplex)
throws RemoteException;
}
2100

FIG. 21

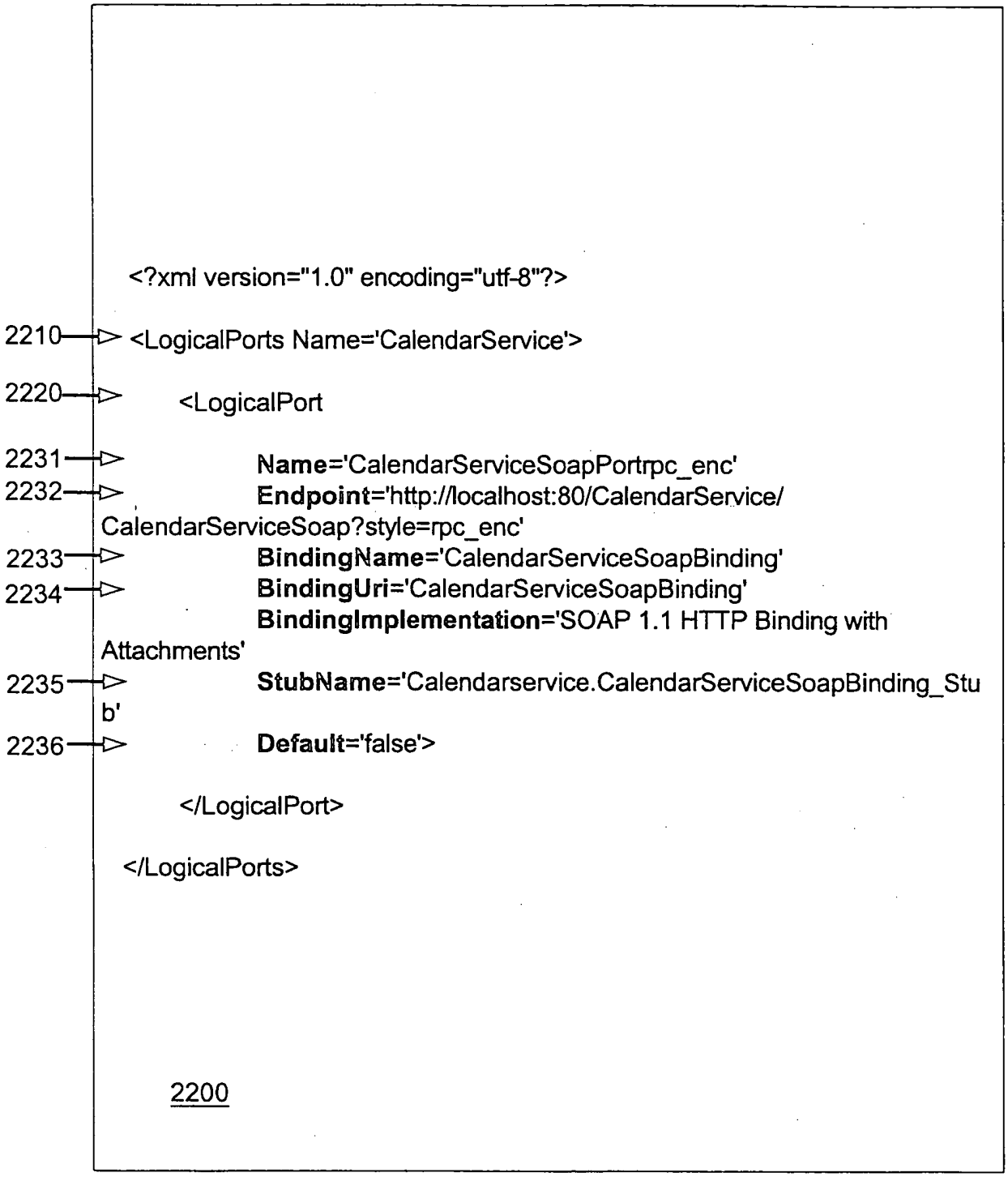

<?xml version="1.0" encoding="utf-8"?>
2210 <LogicalPorts Name='CalendarService'>
2220 <LogicalPort
2231 Name='CalendarServiceSoapPortrpc_enc'
2232 Endpoint='http://localhost:80/CalendarService/
CalendarServiceSoap?style=rpc_enc'
2233 BindingName='CalendarServiceSoapBinding'
2234 BindingUri='CalendarServiceSoapBinding'
BindingImplementation='SOAP 1.1 HTTP Binding with
Attachments'
2235 StubName='Calendarservice.CalendarServiceSoapBinding_Stu
b'
2236 Default='false'>
</LogicalPort>
</LogicalPorts>
2200

FIG. 22

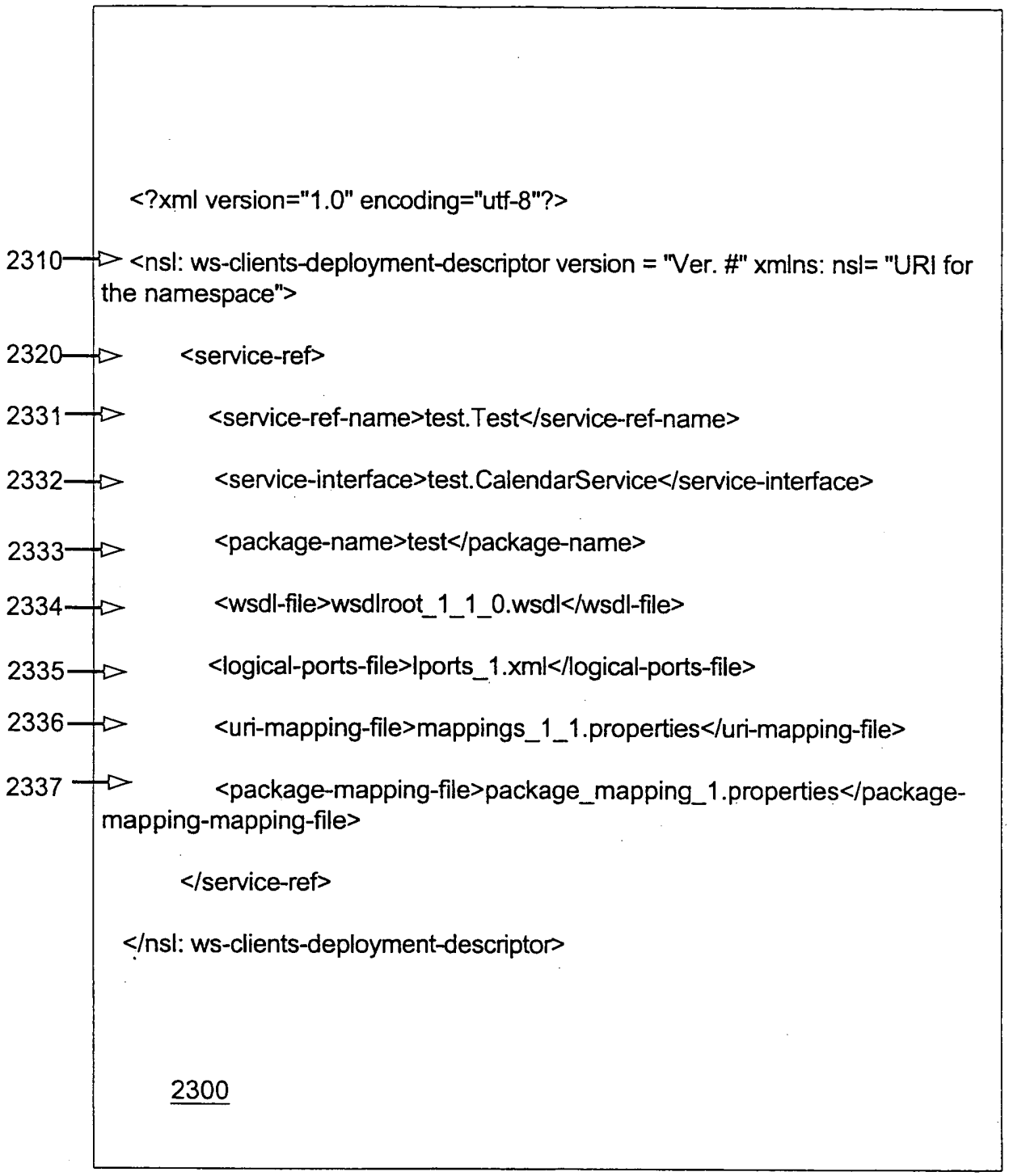

<?xml version="1.0" encoding="utf-8"?>
2310 <nsl: ws-clients-deployment-descriptor version = "Ver. #" xmlns: nsl= "URI for the namespace">
2320 <service-ref>
2331 <service-ref-name>test.Test</service-ref-name>
2332 <service-interface>test.CalendarService</service-interface>
2333 <package-name>test</package-name>
2334 <wsdl-file>wsdlroot_1_1_0.wsdl</wsdl-file>
2335 <logical-ports-file>lports_1.xml</logical-ports-file>
2336 <uri-mapping-file>mappings_1_1.properties</uri-mapping-file>
2337 <package-mapping-file>package_mapping_1.properties</package-mapping-mapping-file>
</service-ref>
</nsl: ws-clients-deployment-descriptor>
2300

FIG. 23

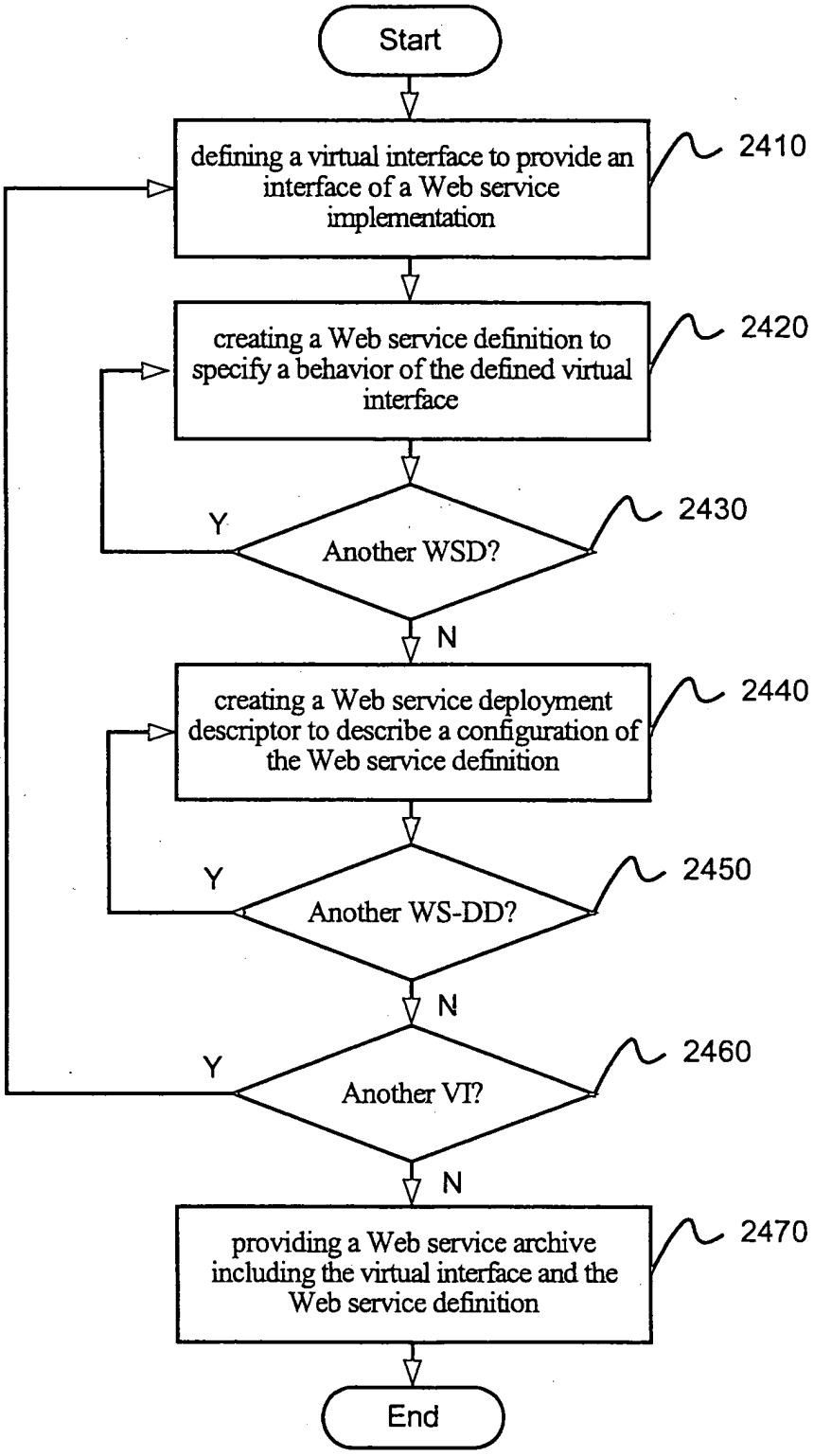

Start
defining a virtual interface to provide an interface of a Web service implementation
2410
creating a Web service definition to specify a behavior of the defined virtual interface
2420
Y
Another WSD?
2430
N
creating a Web service deployment descriptor to describe a configuration of the Web service definition
2440
Y
Another WS-DD?
2450
N
Y
Another VI?
2460
N
providing a Web service archive including the virtual interface and the Web service definition
2470
End

FIG. 24

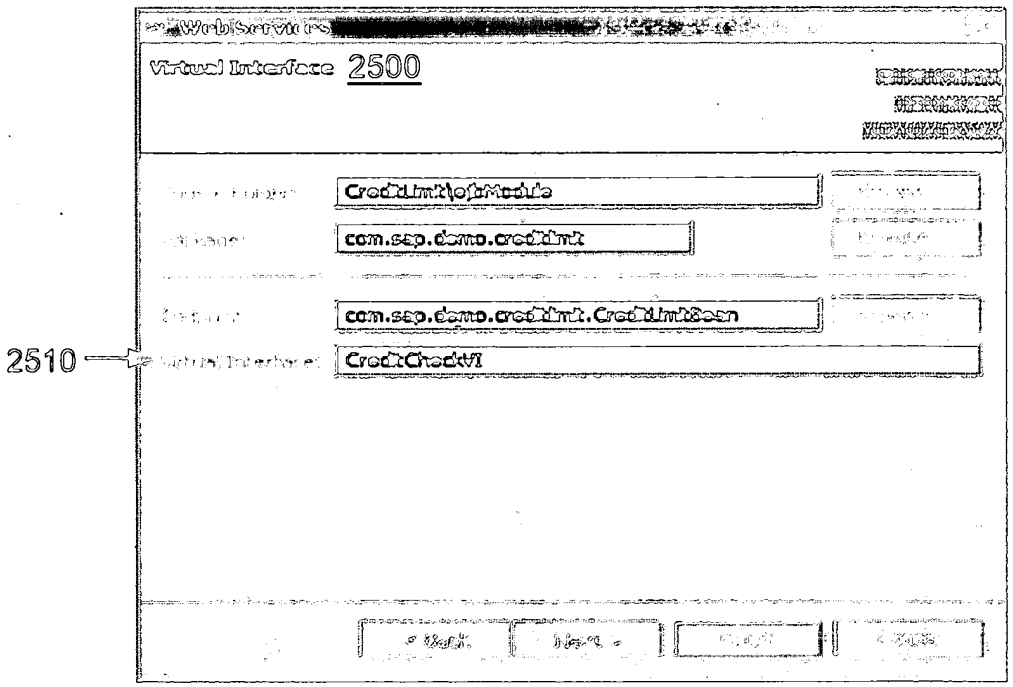


FIG. 25A

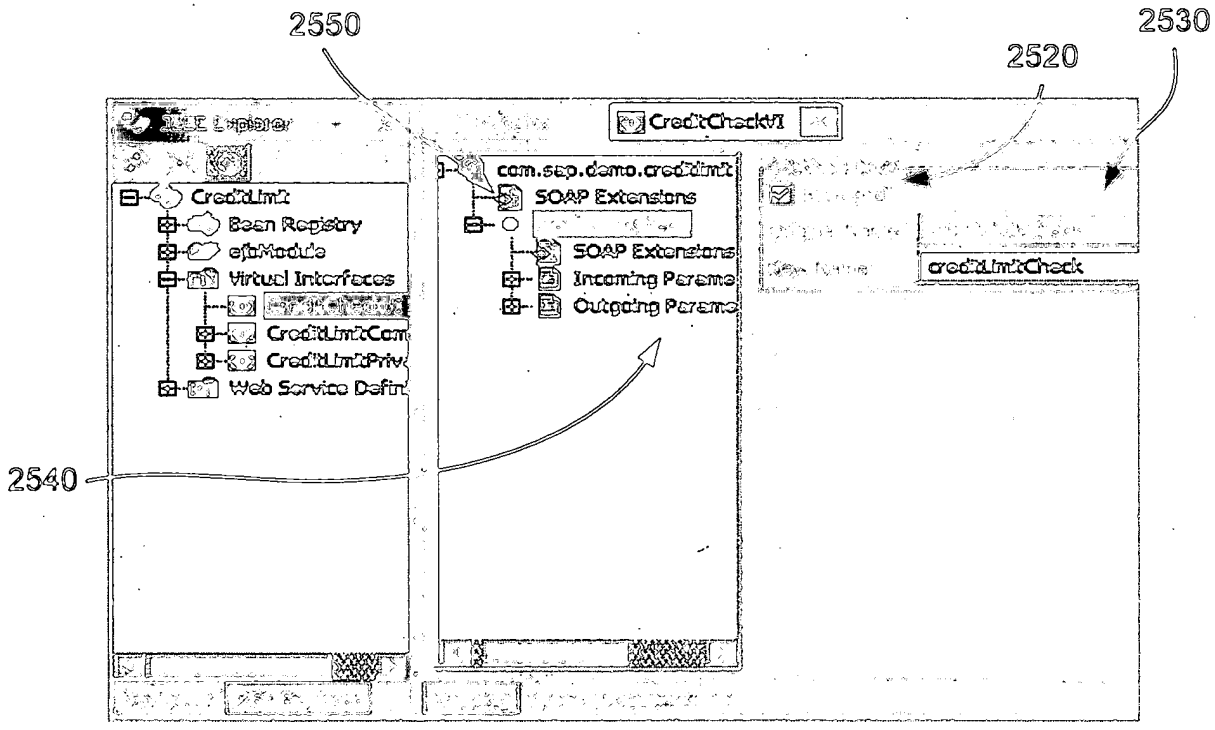


FIG. 25B

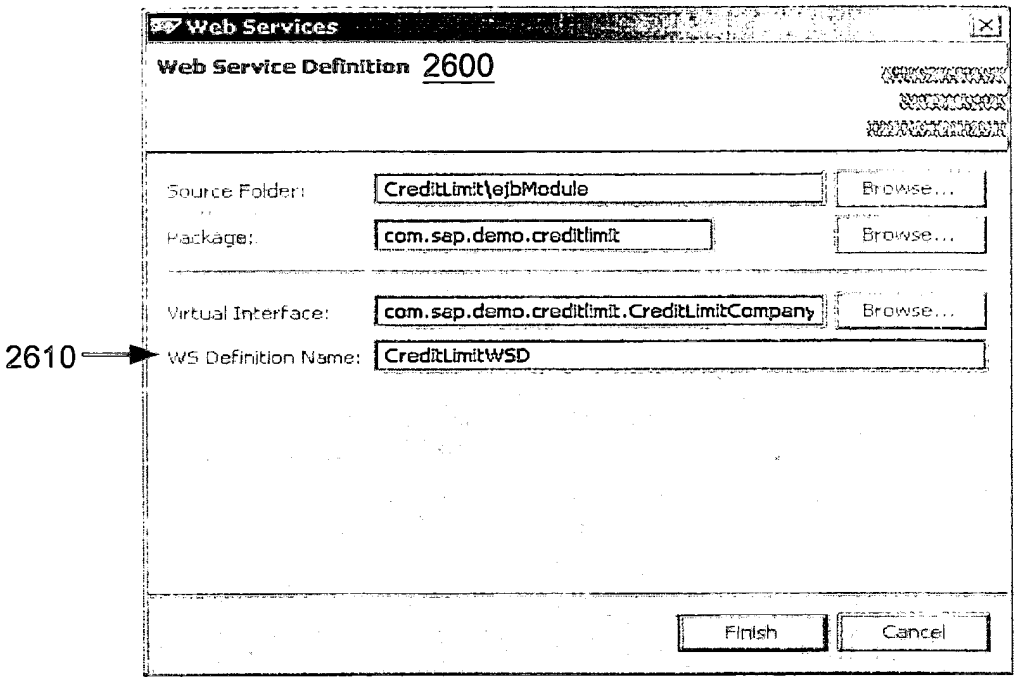


FIG. 26A

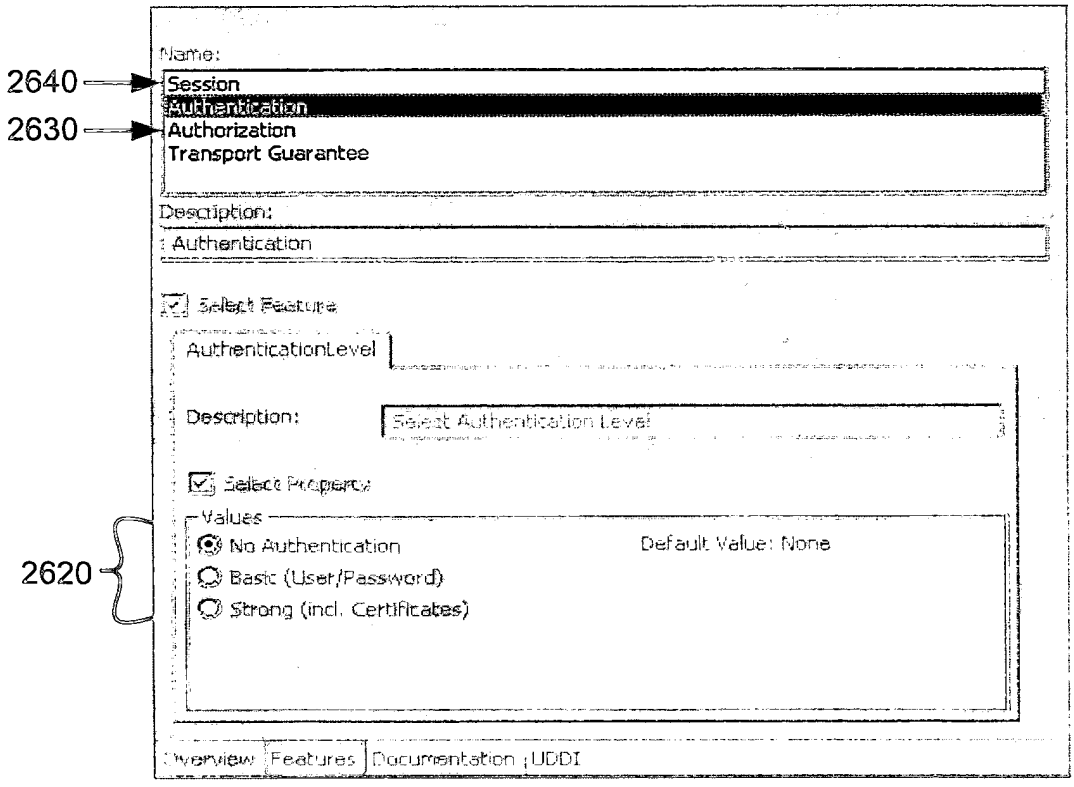
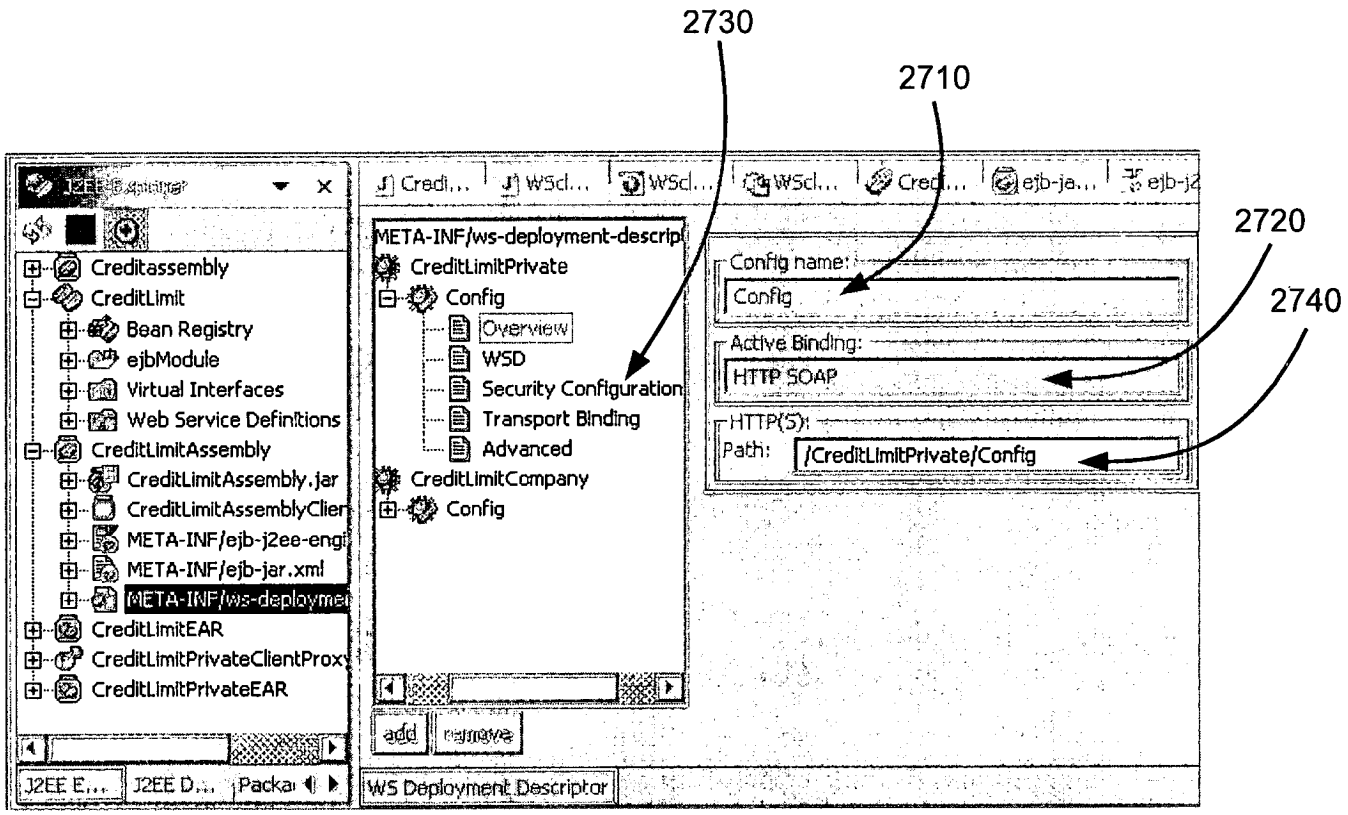


FIG. 26B


2730
2710
2720
2740
Creditassembly
CreditLimit
Bean Registry
ejbModule
Virtual Interfaces
Web Service Definitions
CreditLimitAssembly
CreditLimitAssembly.jar
META-INF/ejb-jar.xml
CreditLimitEAR
CreditLimitPrivateEAR
META-INF/ws-deployment-descrip
CreditLimitPrivate
Config
Overview
WSD
Security Configuration
Transport Binding
Advanced
CreditLimitCompany
Config
Config name:
Config
Active Binding:
HTTP SOAP
HTTP(S):
Path:
/CreditLimitPrivate/Config
add
remove
WS Deployment Descriptor

2700

FIG. 27

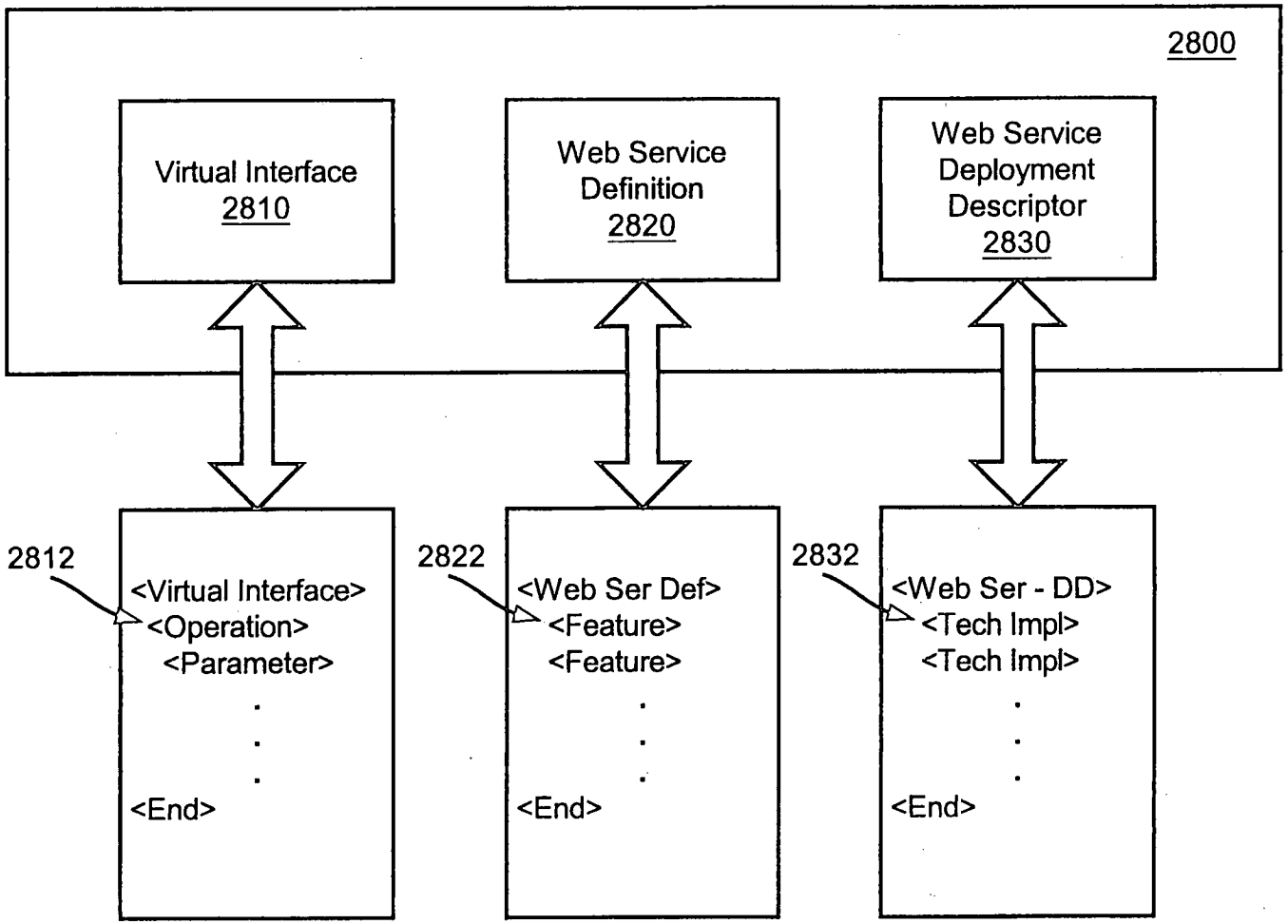

2800
Virtual Interface
2810
Web Service
Definition
2820
Web Service
Deployment
Descriptor
2830
2812
<Virtual Interface>
<Operation>
<Parameter>
<End>
2822
<Web Ser Def>
<Feature>
<Feature>
<End>
2832
<Web Ser - DD>
<Tech Impl>
<Tech Impl>
<End>

FIG. 28

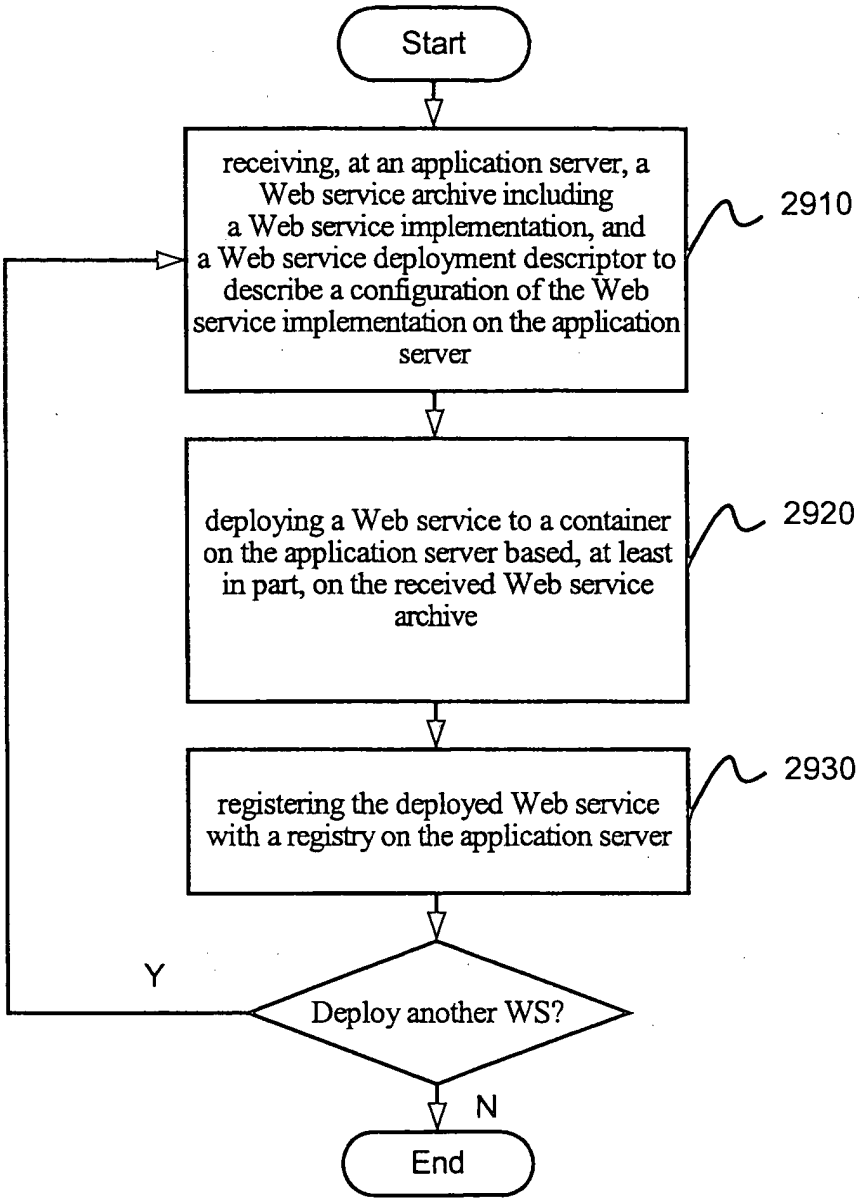

Start
receiving, at an application server, a Web service archive including a Web service implementation, and a Web service deployment descriptor to describe a configuration of the Web service implementation on the application server
2910
deploying a Web service to a container on the application server based, at least in part, on the received Web service archive
2920
registering the deployed Web service with a registry on the application server
2930
Deploy another WS?
Y
N
End

FIG. 29

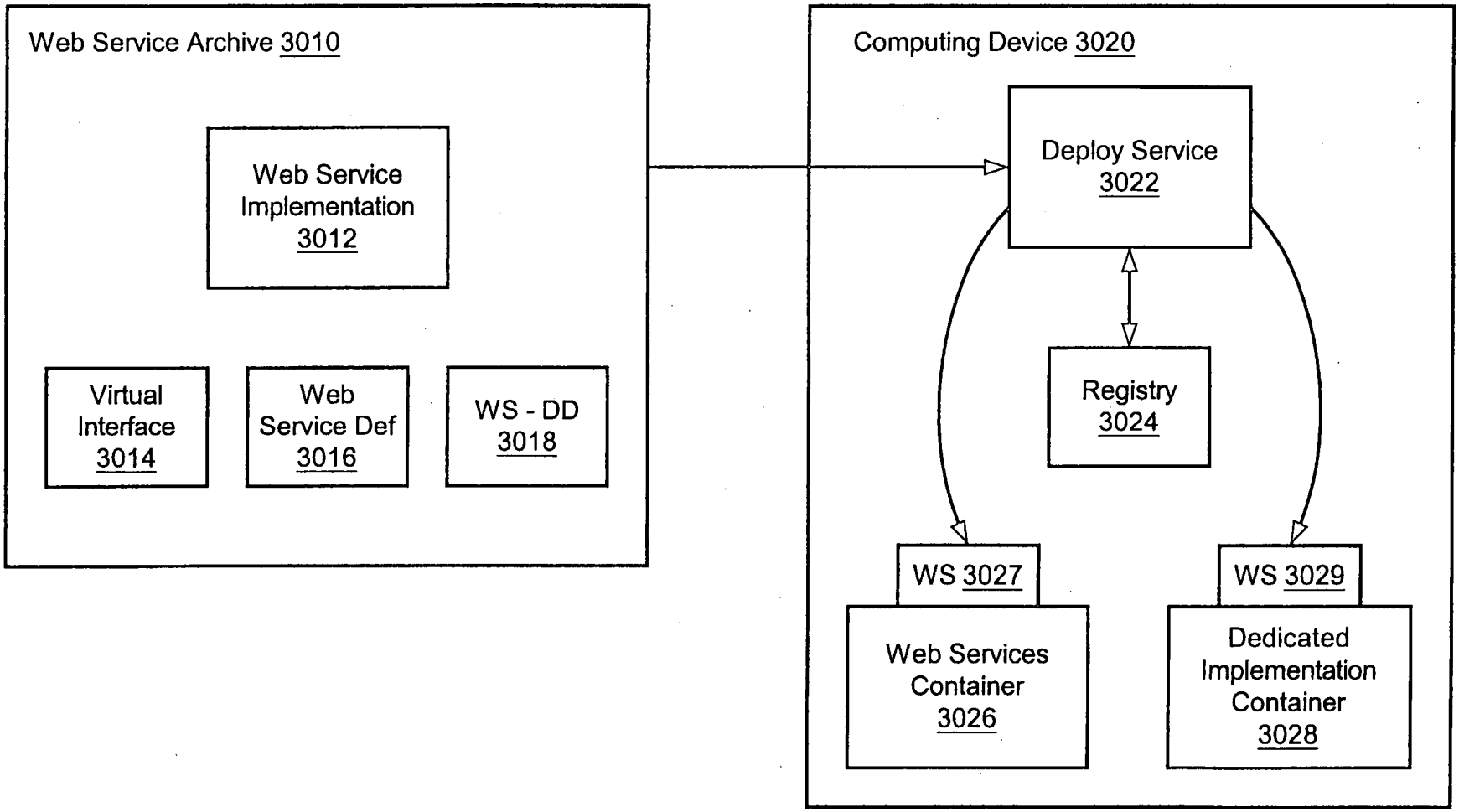

Web Service Archive 3010
Web Service Implementation 3012
Virtual Interface 3014
Web Service Def 3016
WS - DD 3018
Computing Device 3020
Deploy Service 3022
Registry 3024
WS 3027
Web Services Container 3026
WS 3029
Dedicated Implementation Container 3028

FIG. 30

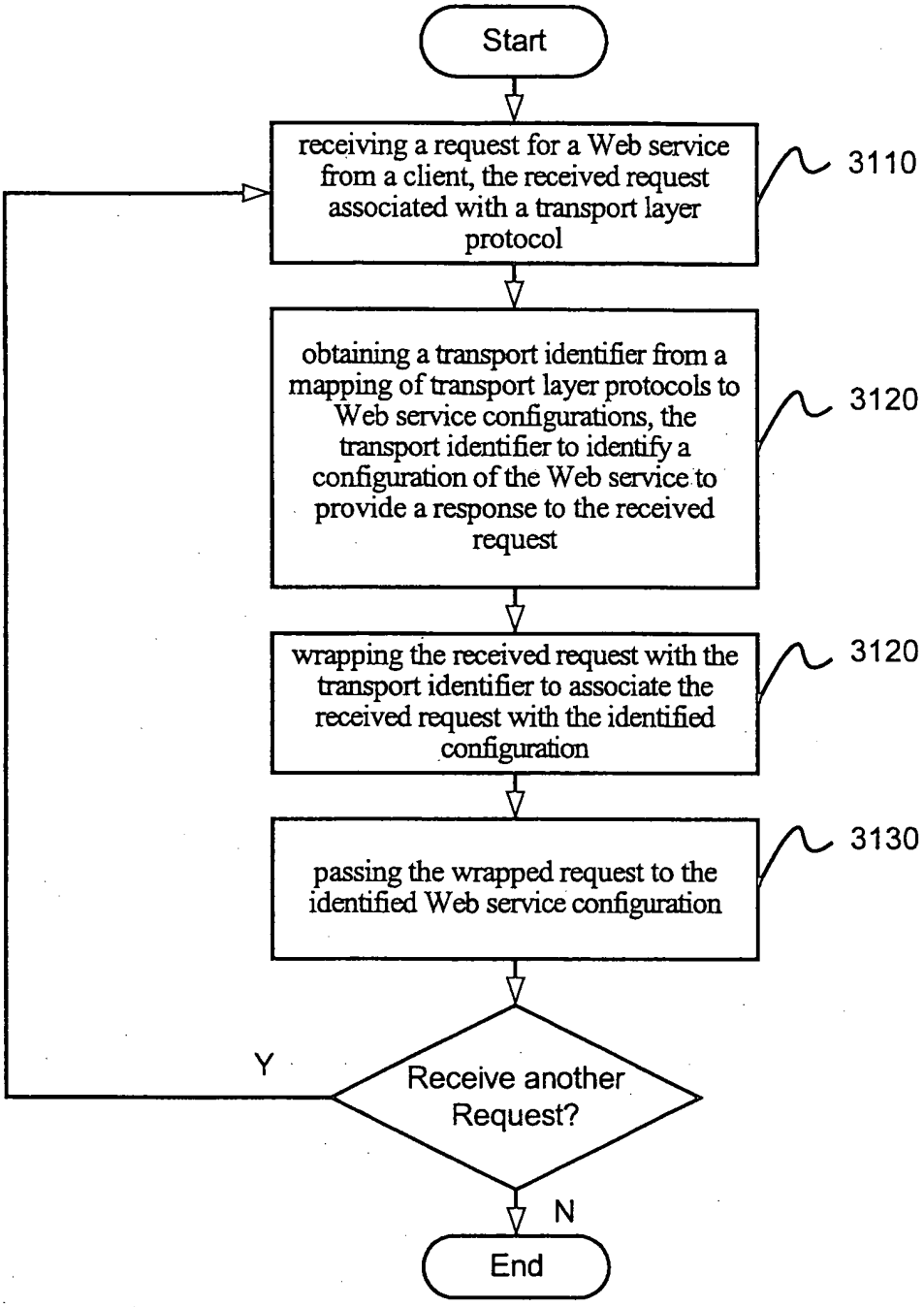

Start
receiving a request for a Web service from a client, the received request associated with a transport layer protocol
3110
obtaining a transport identifier from a mapping of transport layer protocols to Web service configurations, the transport identifier to identify a configuration of the Web service to provide a response to the received request
3120
wrapping the received request with the transport identifier to associate the received request with the identified configuration
3120
passing the wrapped request to the identified Web service configuration
3130
Receive another Request?
Y
N
End

FIG. 31

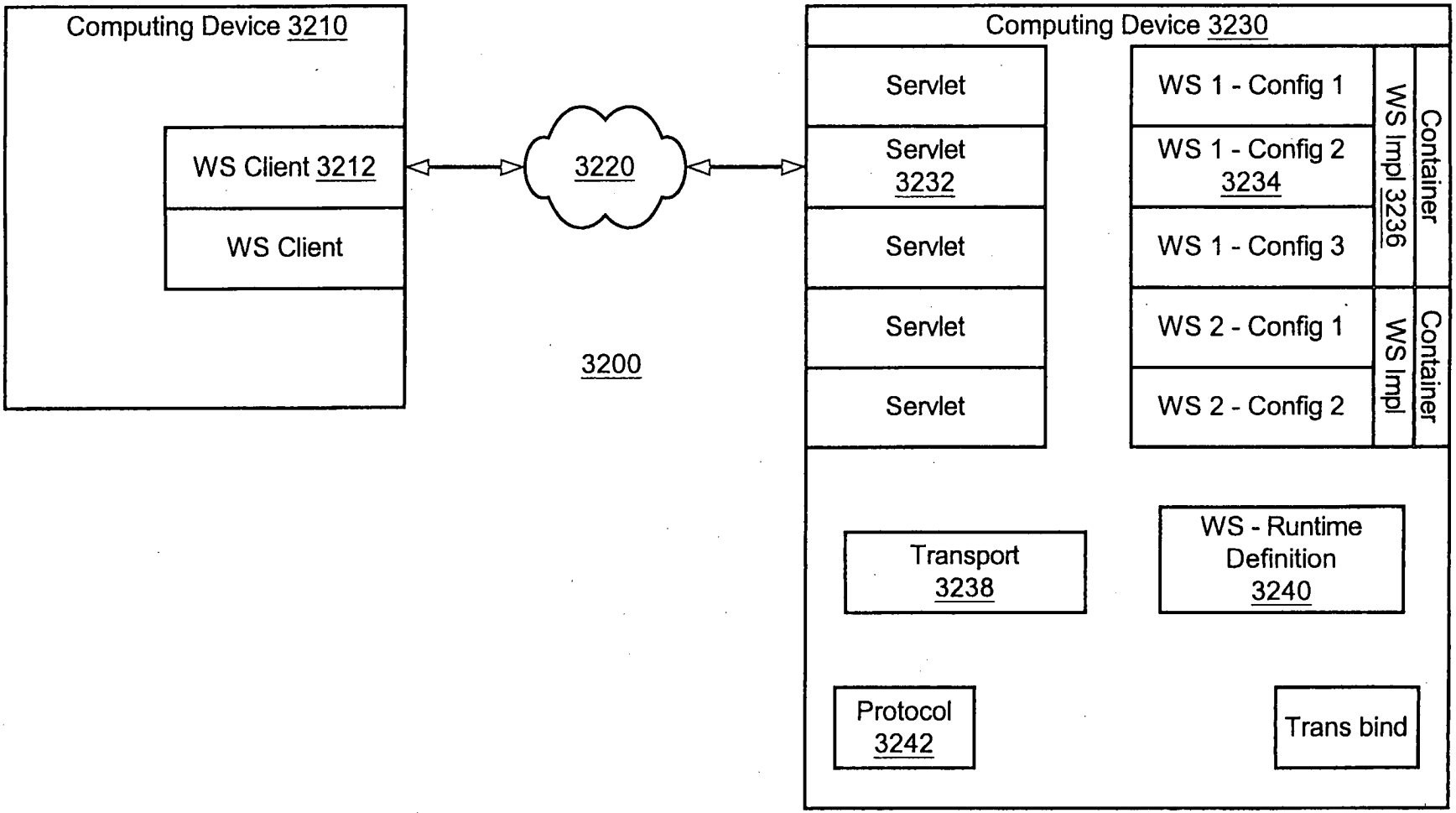

Computing Device 3210
WS Client 3212
WS Client
3220
3200
Computing Device 3230
Servlet
Servlet 3232
Servlet
Servlet
Servlet
WS 1 - Config 1
WS 1 - Config 2 3234
WS 1 - Config 3
WS 2 - Config 1
WS 2 - Config 2
WS Impl 3236
WS Impl
Container
Container
Transport 3238
WS - Runtime Definition 3240
Protocol 3242
Trans bind

FIG. 32

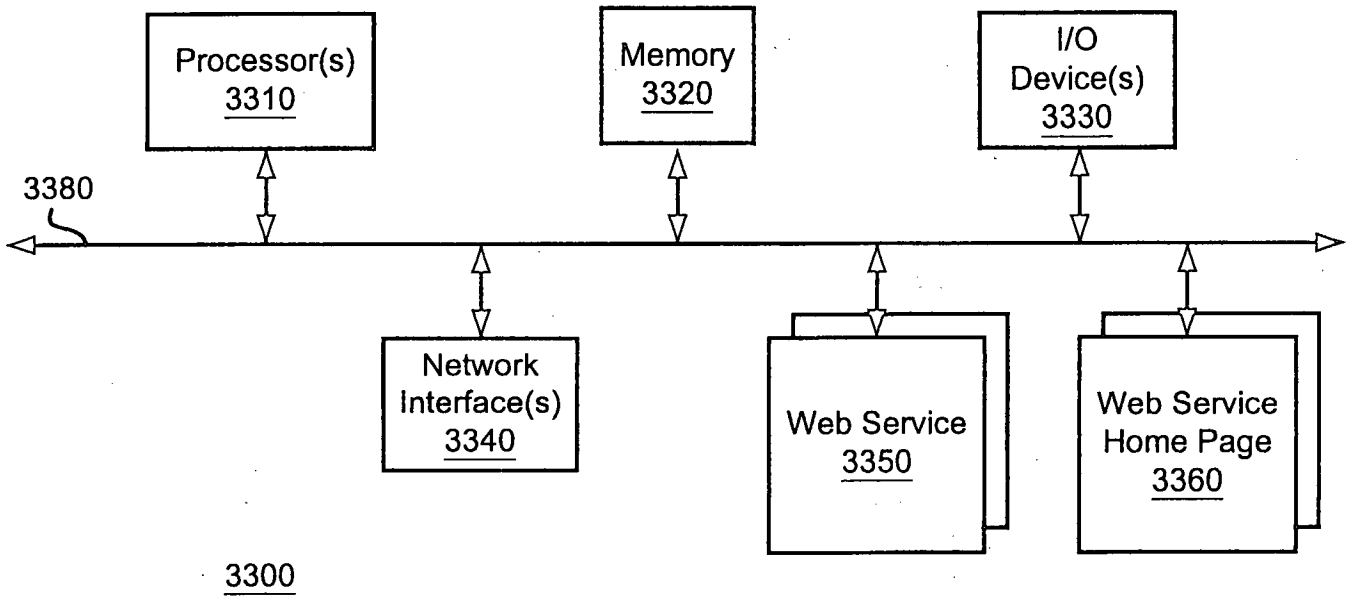

Processor(s)
3310
Memory
3320
I/O
Device(s)
3330
3380
Network
Interface(s)
3340
Web Service
3350
Web Service
Home Page
3360
3300

FIG. 33